INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Moscova

Mobility, Security, Concurrency, Verification and Analysis

Paris - Rocquencourt

THEME COM

Table of contents

1. Team **1**
2. Overall Objectives **1**
3. Scientific Foundations **2**
3.1. Concurrency theory 2
3.2. Type systems 3
3.3. Formal security 3
3.4. Labeled lambda-calculus 3
4. Application Domains **3**
5. Software **3**
5.1. OTT: One true tool for the working semanticist 3
5.2. Caml/Jocaml 4
5.3. Pattern matching in Ocaml 5
5.4. Burfiks: Bayesian web filtering 5
5.5. Hevea 5
5.6. Active DVI 6
6. New Results **6**
6.1. Join-calculus with values and pattern-matching 6
6.2. Concurrent C Minor 6
6.3. Secure Sessions 7
6.4. Models of Audit Logs 8
6.5. Verified Implementations of Cryptographic Protocols 8
6.6. Labeled Lambda-Calculus and Variants 8
6.7. Security Properties in the Lambda-Calculus 9
7. Other Grants and Activities **10**
7.1. National actions 10
7.2. European actions 10
8. Dissemination **10**
8.1. Animation of research 10
8.2. Teaching 11
8.3. Participations to conferences, Seminars, Invitations 11
8.3.1. Participations to conferences 11
8.3.2. Other Talks and Participations 12
8.3.3. Visits 13
9. Bibliography **13**

1. Team

Head of project-team
Jean-Jacques Lévy [Senior Researcher (DR) Inria]

Vice-head of project-team
Luc Maranget [Research Associate (CR) Inria]

Administrative assistant
Sylvie Loubressac [Assistant (AI) Inria]

Research scientists
Ricardo Corin [Research Associate (CR) Inria, since 1/10/2007]
James Leifer [Research Associate (CR) Inria]
Francesco Zappa Nardelli [Research Associate (CR) Inria]

Post doctorant
Louis Mandel [postdoc INRIA, to 30/09/2007]

Ph. D. students
Jade Alglave [MESR grant, Paris 7, from 1/10/2007 following 3 month internship]
Pierre-Malo Deniélou [AMN, Paris 7]
Nataliya Guts [INRIA-MSR grant, Paris 7, since 1/10/2007 following 3 month internship]

Student intern
Arun Kumar Patala [IIT Kanpur, from 05/10/2006 to 07/30/2006]
Vidyut Ghosal [IIT Kanpur, from 05/10/2006 to 07/30/2006]

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 [47], Obliq, Nomadic Pict [43], 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 [15] prototype, developed jointly at U. of Cambridge and at INRIA, demonstrates 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 2006: 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 2007, Gilles Peskine (presently with P. Sewell, Cambridge) is now ready to defend his PhD (which has been reviewed in December 2007). Louis Mandel finished a post-doctoral year in our project-team (he has been recruited at University of Orsay); Arun Kumar Patala and Vidyut Ghosal (IIT Kanpur) were interns with F. Zappa Nardelli. Jade Alglave and Nataliya Guts, interns of MPRI, started a PhD programme in Moscova.

Since August 2006, J.-J. Lévy is also director of the new 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 [40]. 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 [41] 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 [3], [4], 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 [42] 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.

3.4. Labeled lambda-calculus

The theory of Church lambda-calculus is considered in our work about dynamic access control to library functions. More specifically, the theory of the confluent history-based calculus [39] defines a labeled lambda-calculus which is used both for access control in libraries and for the reversible pi-calculus.

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 [27], [26] 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.

Our main goal 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.

Most simply, the tool can be used to aid completely informal LaTeX mathematics. Here it permits the definition, and terms within proofs and exposition, to be written in a clear, editable, ASCII notation, without LaTeX noise. It generates good-quality typeset output. By parsing (and so sort-checking) this input, it quickly catches a range of simple errors, e.g. inconsistent use of judgment forms or metavariable naming conventions.

That same input can be used to generate formal definitions, for Isabelle, Coq, and HOL. It should thereby enable a smooth transition between use of informal and formal mathematics. Additionally, the tool can automatically generate definitions of functions for free variables, single and multiple substitutions, sub-grammar checks (e.g. for value sub-grammars), and binding auxiliary functions.

Our focus here is on the problem of writing and editing language definitions, not (directly) on aiding mechanized proof of meta-theory. If one is involved in hard proofs about a relatively stable small calculus then it will aid only a small part of the work (and one might choose instead to work just within a single proof assistant), but for larger languages the definition is a more substantial problem — so much so that only a handful of full-scale languages have been given complete definitions. We aim to make this more commonplace, less of a heroic task.

In collaboration with Peter Sewell (Cambridge University).

The current version of Ott is about 20000 lines of OCaml. It is available from
http://moscova.inria.fr/~zappa/software/ott (BSD licence).

5.2. Caml/Jocaml

Participants: Louis Mandel, 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.

The basic idea of JoCaml is to promote the linguistic approach to concurrent programming. By constrast to the library based approach, the integration of concurrent constructs into the programming language offers more opportunies for checking the programmer's code (*e.g.* expressive type checking), optimizing it, and restricting its freedom to write incorrect code. Another, apparently minor, but important in practice, advantage is a convenient syntax close to the underlying program abstraction (in our case the join-calculus). Hence programmers are invited to think in terms of the underlying semantics, and therefore in terms of abstraction at the design time. In other words, we believe in "high-level" programming languages for concurrent programming.

Finally, acting from inside the compiler also permits tight interaction between the various features of the programming language. For instance, we offer deep integration of ML pattern matching and join matching.

The first public release of JoCaml (3.10, so as to match Objective Caml release numbers) happened in June 2007. This release [30] includes:

- The system with bytecode and native code compilers and runtime environments, toplevel, and basic libraries. The`jocaml` system is a modified `ocaml` system, of which about ten thousand lines of source code are modified or added. In pratice the sources of `jocaml` reside in a branch of ocaml development sources and maintenance is easily performed.
- A comprehensive documentation, written by L. Maranget and L. Mandel.
- Additional material, such as a mailing list interface, related articles and a few programming examples.

The next objective is to develop a community of users programming in JoCaml. We saw some interest for JoCaml at the release time. One programmer, not from our team, used JoCaml is a small permanent programming competition (Wide Finder) achieving good performance http://eigenclass.org/hiki.rb?fast-widefinder.

To provide potential programmers with realistic examples, we ourselves wrote a middle-sized program (a few thousand lines): a distributed ray tracer. We wrote a research report on this experience [28] and submitted a tool presentation for the ESOP'08 conference, which is accepted [24]. The ray tracer achieves good performance. Nevertheless our main 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.

5.3. Pattern matching in Ocaml

Keywords: *ocaml*, *pattern matching*.

Participant: Luc Maranget.

This work examines the ML pattern-matching anomalies of useless clauses and non-exhaustive matches. Providing good diagnoses for these anomalies helps enormously programmers: the novices better understand the subtleties of ML pattern-matching, the experienced programmers better control the effect of program modifications.

We state the definition of these anomalies, building upon pattern matching semantics, and propose a simple algorithm to detect them. This algorithm has been integrated in the Objective Caml compiler by Luc Maranget. The same algorithm is also usable in non-strict languages such as Haskell. Or-patterns can be considered for both strict and non-strict languages.

After almost two years of refereeing, this paper (which was accepted last year) is now published in the "*Journal of Functional Programming*" [14].

5.4. Burfiks: Bayesian web filtering

Participants: Arun Kumar Patala, Vidyut Ghosal, Francesco Zappa Nardelli.

This project studies and implement a statistical approach to improve the quality of the results returned by web search engines, and more generally, to sort the links of a web page according to their interest.

An inevitable consequence of the ambiguity of the natural language used to express search queries is that interesting results of searches on the web are often hidden among many unrelated matchings. The use of endogenous webs (among the documents that match a query, the most relevant is the one which is pointed by the greatest number of them) as done by Google turned out to be a promising solution.

However, as far as we know, an interesting and simple approach has not been considered yet: search by Bayesian filtering. Bayesian filters have been made popular by Paul Graham to identify automatically Spam messages [36]. They rely on an elementary theorem of probability theory known as "Bayes formula". Intuitively, by examining old messages that have been prealably classified by the user into "Spam" and "not-Spam", they build a probability distribution that has good properties to predict if an incoming message will be classified by the user as Spam or not.

This project aims at applying the Bayesian approach to refine the results returned by web searches, and, more generally sorting the links found in a web page according to their interest.

We developped a software layer integrated with the Firefox browser that collects the links found in a web page and uses hints of the user to classify them, along the lines of anti-Spam Bayesian filters. Preliminary experience revealed that this tool is extremely useful to navigate quickly across pages that contains many links, as it identifies quite reliably the most interesting ones in real time.

An alpha release of the software is available from http://burfiks.gforge.inria.fr (GPL licence).

In collaboration with Roberto Di Cosmo (PPS, U. Paris 7).

5.5. 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 the release of Hevea 1.10. Main improvements over previous version 1.09 are:

- Hevea now accepts UTF-8 encoded files — *i.e.* it features an implementation of LaTeX `utf8` and `utf8x` packages.
- The `natbib` package (a popular package for bibliographies) is implemented.
- All bugs reported by users are corrected.

Hevea is distributed as free software from its own web site http://hevea.inria.fr. The audience of Hevea is difficult to estimate, given the well established practice of binary packaging by several independent Linux distributions. As an indication of popularity, more than 2000 installations of `hevea` have been reported for Debian and Ubuntu Linux distribution. Other linux distributions such as Mandrake also feature `hevea`.

5.6. Active DVI

Keywords: *dvi*, *latex*, *tex*.

Participant: Francesco Zappa Nardelli.

`Advix` is a new implementation of ActiveDVI, a programmable viewer for DVI files developed by the Cristal project-team at INRIA Rocquencourt. It improves the original one by (among other things) providing faster rendering of postscript specials, and by being compatible with Windows and MacOs. A pre-alpha version is being tested (about 48000 lines of OCaml and C code).

In collaboration with Didier Remy (INRIA Rocquencourt).

6. New Results

6.1. Join-calculus with values and pattern-matching

Participants: Luc Maranget, Ma Qin.

This 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.

We have submitted this work to the Journal *Logical Methods in Computer Science* in January. At the time of this report, the paper is "accepted with minor revisions" and we have submitted the revised version.

6.2. Concurrent C Minor

Participants: Jade Alglave, Andrew Appel [U. Princeton], Aquinas Hobor [U. Princeton], 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 optimizing 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 [38].

In some application domains it is not enough to build reliable software systems, one wants proved-correct software. Since proofs are large and complex, the proof-checking must be mechanized. Machine-checked proofs of real software systems are difficult, but now should be possible, given the recent advances in the theory and engineering of mechanized proof systems applied to software verification. But there are several challenges:

- Real software systems are usually built from components in different programming languages.
- Some parts of the program need full correctness proofs, which must be constructed with great effort; other parts need only safety proofs, which can be constructed automatically.
- One reasons about correctness at the source-code level, but one runs a machine-code program translated by a compiler; the compiler must be proved correct.
- These proofs about different properties, with respect to different programming languages, must be integrated together end-to-end in a way that is also proved correct and machine-checked.

We address these challenges by defining 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 [23], [16] a Concurrent Separation Logic [44] 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 formalize and reason about relaxed memory models.

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

In collaboration with Sandrine Blazy (ENSIIE and INRIA Rocquencourt), and Adriana Compagnoni (Stevens Tech.).

6.3. Secure Sessions

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

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.

This work was presented at CSF'07 [21], TGC'07 [20] and will appear in a special issue of the Journal of Computer Security [11]

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 analyzed 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, we formalize audit logs for a sample optimistic scheme, the value commitment [22]. It is specified in a pi calculus extended with committable locations, and compiled using standard cryptography to implement secure logs. We show that our distributed implementation either respects the semantics of commitments or, using the information stored in the logs, detects cheating by a hostile environment.

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. So far we investigated two example protocols: an implementation of the Otway-Rees protocol, and an implementation of a simplified password-based authentication protocol. In both cases, we obtained concrete security guarantees for a model closely related to executable code. In the future, we would like to consider the more challenging case of SSL/TLS, a real life protocol used pervasively.

As an alternative to CryptoVerif, we are considering also developing proofs in a computational cryptography model built in the Coq theorem prover (ongoing work with Sophia-Antipolis).

Preliminary work appears at FCC07 [18], [19].

6.6. Labeled Lambda-Calculus and Variants

Participants: Tomasz Blanc, Jean-Jacques Lévy, Luc Maranget.

We introduced a new property of the labeled lambda-calculus: *context irreversibility*. We have $C[M] \to C[M']$ if and only if $M \to M'$. This property shows that when a (labeled) context disappears at one point of a reduction, this disappearance is irreversible: the context cannot be rebuilt in the reduction that follows. In the (unlabeled) lambda-calculus, this property is false: we have $(\lambda x.xy)y \to yy$ and $\lambda x.xy\neg \to y$. Lévy had put in light in his thesis that syntactic coincidences might occur in the lambda-calculus [39]. These coincidences are responsible for the loss of context irreversibility in the unlabeled calculus. Context irreversibility is similar to time irreversibility: labeled contexts can be used to time-stamp a reduction.

We examined two variants of the lambda-calculus. In the lambda-calculus by value, only redexes whose right part is a value (i.e. an abstraction) are contracted. The labels express the property of stability in the lambda-calculus but not in the lambda-calculus by value. We adapted the labels to recover this property. The lambda-calculus by value and the labeled lambda-calculus by value are confluent and verify the theorems of finite developments and standardization (although the definition of a standard reduction is adapted). To prove finite developments, we used an elegant, intuitive technique based on a notion of *future redex imbrication*.

We also examined the weak lambda-calculus. Although its syntactic properties did not receive great attention in the past decades, this theory is more relevant for the implementation of programming languages than the usual theory of the strong lambda-calculus. Contrary to the latter and similarly to lazy functional languages, the weak lambda-calculus does not validate the ξ-rule i.e. the reduction under a lambda-abstraction. Without this rule, the weak lambda-calculus is not confluent. We based our labeled language on a variant of the weak lambda-calculus by the Çağman and Hindley for Combinatory Logic [48]. In this variant, a new ξ'-rule is valid: it allows reductions under lambda-abstraction in sub-terms that do not contain a bound variable. This variant enjoys confluence. We proposed a labeling of this language that expresses a confluent theory of reductions with sharing, independent of the reduction strategy. If the sub-terms of a term are initially labeled with distinct letters, then, after some steps of reduction, two sub-terms that have the same label are equal. This formal setting corresponds to the shared evaluation strategy by Wadsworth defined with dags in 1971. This work was published on the occasion of Jan Willem Klop's 60th birthday [33] in 2005. A slightly simplified version of this work, context irreversibility and the labeled lambda-calculus by value are presented in Tomasz Blanc's thesis [32] in 2006. This year, we presented a simplified version of this work at the symposium for Henk Barendregt's 60th birthday [10].

6.7. Security Properties in the Lambda-Calculus

Participants: Tomasz Blanc, Jean-Jacques Lévy.

In extensible virtual machines such as the JVM or the CLR, software components from various origins (applets, local libraries, ...) share the same local resources (CPU, files, ...) but not necessarily the same level of trust. We model access control in such semi-trusted environment in a minimal language based on the lambda-calculus. We aim at expressing in this framework a variety of known access control mechanisms.

The stack inspection is a dynamic access control mechanism that is used in the JVM and the CLR. Before accessing a sensitive resource, the call stack is inspected to check that every caller is allowed to access the resource. This mechanism is difficult to handle since it depends on the runtime stack, which is not statically known. Moreover, Fournet and Gordon showed that it is surprisingly hard to express what security property is guaranteed by stack inspection [35]. With static analysis, one can guarantee statically that such security defects will not occur [1] [37].

Flow analysis is another approach with stronger theoretical foundations. This analysis consists in tracking secret data along the execution of a program: these data are not accessible to untrusted code [46]. This approach has two advantages: (1) it is particularly adapted to static analysis and (2) it formally guarantees a security property: non-interference.

Nevertheless a static flow analysis is insufficient to express inherently dynamic security policies such as the Chinese Wall policy which is inspired by interest conflicts [34], [45]. This policy is defined as follows: initially, Alice may choose to communicate with Bob or Charlie; but once she communicated with Bob (resp. Charlie), she is not allowed anymore to communicate with Charlie (resp. Bob).

All these security mechanisms rely crucially on history of past events. To trace this history in the lambda-calculus, we use Lévy's labels: the labels of redexes keep track of the past interactions that created it [39]. We introduced a variant of the labeled lambda-calculus where the contraction of a redex is conditioned by the name of the redex and by its context. We proved that an instance of this language corresponds to a reasonable variant of the lambda-calculus with stack inspection that was proposed by Fournet and Gordon in [35].

We examined the property of non-interference in the lambda-calculus and in the lambda-calculus by value. A sub-term of an initial term M interferes if it influences the result of M's reduction. We showed that this property is strongly related with the stability property. In a lambda-calculus with references, in addition to the "functional" interference that already exists in the pure lambda-calculus, there is a "memory" interference due to the fact that the use of a reference may influence the result of a reduction. More precisely, the content of a reference during a time interval may have an influence. We adapted the labels of the lambda-calculus to capture this phenomenon: we took advantage of context irreversibility property to time-stamp reductions. In this labeled calculus, we formally expressed (non-)interference.

We introduced principals in the lambda-calculus. In this calculus, we defined a new security property: independence. A reduction $\mathcal{R}$ involving two principals A and B is independent from the interaction between A and B if this reduction may be decomposed into two reductions $\mathcal{R}_A$ and $\mathcal{R}_B$ where $\mathcal{R}_A$ (respectively $\mathcal{R}_B$) ignores the principal A (resp. B). To express the Chinese Wall policy, we took advantage of the labels of the lambda-calculus. We proved that a reduction that respects the Chinese Wall policy for Alice and Bob is independent from the interaction between Alice and Bob. This work is presented in Tomasz Blanc's thesis [32].

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 *Provable Secure Distributed Computing* and deals with security, programming languages theory and formal proofs. This work was still under active collaboration within all year 2007.

8. Dissemination

8.1. Animation of research

R. Corin co-organized, with T. Rezk (INRIA Sophia-Antipolis), the Workshop on the Interplay of Programming Languages and Cryptography, November 7, 2007. This workshop brought expert invited speakers leading the field both from Europe and U.S.

R. Corin took part of the Programme Committee of the 3rd International Conference on Availability, Reliability and Security (IEEE ARES 2008).

P.-M. Deniélou participated in the French Science Fest ("Fête de la Science").

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.

J.-J. Lévy co-organized, with Y. Bertot, G. Huet, G. Plotkin, the Gilles Kahn Colloquium, Jan 11, 2007, in memory of our ex INRIA president.

J. Leifer is member of the post-doctoral hiring committee.

L. Maranget is elected member of *Comité technique paritaire* of Inria, 1 meeting per month about the general politics of Inria.

L. Maranget acted in the program committee of the 2007 ML Workshop, affiliated to the ICFP conference.

8.2. Teaching

Our project-team participates to the following courses:

- "Bases de la programmation et de l'algorithmique", 1st year course at the Ecole polytechnique, 75 students, 9×1.5 hours + the final examination, Nov. 2007-Jan. 2008 (L. Maranget is the professor "chargé de cours", in charge of this course, with the help of 4 instructors for the laboratory classes; see the lecture notes [31], 219 pp.);
- "Introduction to Programming Languages Theory", 3rd year course at the Ecole polytechnique, 14 students, 9×1.5 hours for classes and 9×2 hours for laboratory classes, plus the final examination;
- "Concurrency", Master Parisien de Recherche en informatique (MPRI), 2006-2007, at U. of Paris 7, 30 students, (F. Zappa Nardelli taught the pi-calculus semantics: 21 hours and the final examination; one course was done by J. Leifer;
- "Projet Programmation (IK3)", 2nd year course at U. of Paris 7, 1 group (15 students, 4 hours a week) among 70 students (P.-M. Deniélou did the lab classes on Java programming).

We also had the following activity related to teaching:

- J.-J. Lévy co-authored 3 computer science problems (4h+2h+2h) of the entrance examination at the Ecole polytechnique in 2007. He is the coordinator of the Computer Science examinations at École polytechnique.

8.3. Participations to conferences, Seminars, Invitations

8.3.1. Participations to conferences

- Jan 16, J. Leifer attended the FORMACRYPT workshop ("bridging the gap between the formal and computational models of cryptography") organised by Bruno Blanchet at ENS, Paris.
- Jan 17-19, R. Corin, P.-M. Deniélou, J. Leifer, J.-J. Lévy attended POPL 2007 in Nice, and a series of technical meetings on session cryptography with researchers from Microsoft Research Cambridge and INRIA Sophia-Antipolis.
- Feb 2-4, 2-4, J.-J. Lévy and F. Zappa Nardelli attended the kick-off meeting of the ANR ParSec Project. They gave two talks on "History Based Flow Analysis in the lambda calculus" and on "The Concurrent C-minor Project".
- Feb 12-14, F. Zappa Nardelli visited the Computer Laboratory of the University of Cambridge for collaboration with Peter Sewell.
- Feb 16, F. Zappa Nardelli gave a talk on "Tool support for the working semanticist" at the Démons-Proval-Logical seminar in Orsay.
- Mar 2-5, J.-J. Lévy visited the department of Computer Science at University of Chicago.
- Mar 6-9, J.-J. Lévy attented the Microsoft Research TechFest at Seattle.
- Mar 26-30, J.-J. Lévy attended Etaps in Braga.
- Apr 4, J.-J. Lévy presented the research of the Joint Centre at the students of the Corps des Télécommunications, at ENST, Paris.
- Apr 24-26, all Moscova members participated to the evaluation seminar of INRIA ComC Programme, in Paris.
- May 16, J.-J. Lévy visited the Joint Centre between University of Trento and Microsoft Research Cambridge (Corrado Priami) in Trento.
- May 23-24, J.-J. Lévy visited the Machine Learning Group at Microsoft Research Cambridge with J.-D. Boissonnat and F. Chazal.

- May 3, L. Maranget presented a talk on ML pattern maching at the third seminar of the ARC Quotient (May 3 at LIP6, Paris).
- May 7-10, L. Maranget attended the colloquium "Algorithmique et Programmation" at the CIRM Marseille Luminy. He presented 3 lectures on ML pattern-matching (3.5 hours) to a selected audience of teachers in "*classe préparatoires*".
- Jun 20, J. Alglave gave a talk on "Certified symbolic execution with separation logic" at the 2nd ANR ParSec meeting in Orsay.
- Jun 22, J.-J. Lévy gave at talk at the Symposium in Honnour of the 60th anniversary of Gérard Huet.
- Jun 26-29, J.-J. Lévy attended the RDP'07 conference in Paris.
- Jul 6-8, N. Guts attended the "20th IEEE Computer Security Foundations Symposiom" in Venice, Italy. She gave a short talk on "Formal properties of audit logs".
- Jul 6-8, R. Corin, P.-M. Deniélou, J. Leifer attended the "20th IEEE Computer Security Foundations Symposium", where our paper [21] was presented, and for the Workshop on Formal and Computational Cryptography FCC 2007.
- Jul 9-10, J.-J. Lévy attended the Symposium celebrating the 10th anniversary of the Microsoft Research laboratory in Cambridge.
- Jul 17-22, J. Alglave attended the "Separation Fest" meeting at Queen Mary University, London (Royaume-Uni).
- Jul 22-25, F. Zappa Nardelli visited the Computer Laboratory of the University of Cambridge for collaboration with Peter Sewell.
- Sep 27-28, J.-J. Lévy attended the meeting of the Scientific Council of the Fondation de Mathématiques de Paris + the official opening at the Collège de France.
- Sep 30-Oct 4, J. Alglave, J.-J. Lévy, L. Maranget, F. Zappa Nardelli attended the International Conference on Functional Programming (ICFP), Freiburg, Germany.
- Oct 2, F. Zappa Nardelli gave a talk on "Ott: Tool support for the working semanticist" at the "Groupe de travail Moscova/Gallium".
- Oct 11, J.-J. Lévy gave a talk at the Médiathèque du Drancy at the occasion of the Fête de la Science.
- Oct 16-18, F. Zappa Nardelli visited the Computer Laboratory of the University of Cambridge for collaboration with Peter Sewell.
- Oct 17, J.-J. Lévy gave a talk at ENSHEEIT (Toulouse) on the celebration of its 100th anniversary.
- Nov 5-7, R. Corin, P.-M. Deniélou, J. Leifer, J.-J. Lévy attended the Trustworthy Global Computing Conference, where P.-M. Deniélou presented [20], and for the Workshop on the Interplay of Programming Languages and Cryptography, a workshop co-organised by R. Corin.
- Nov 9, N. Guts gave a talk on "A Formal Implementation of Value Commitment" at the Gallium-Moscova seminar, Rocquencourt.
- Dec 10-11, all Moscova members attended the Symposium in Lille for the 40 years of INRIA.
- Dec 13, J.-J. Lévy and F. Zappa Nardelli attended the ANR Parsec meeting. F. Zappa Nardelli gave a talk on OTT.
- Dec 17, J.-J. Lévy presented a talk on the Weak Lambda Calculus at the 60th anniversary of Henk Barendregt, Nijmegen.

8.3.2. Other Talks and Participations

- Jan 22, L. Mandel gave a talk at the LACL group, Univ. Paris 12, Créteil.
- Feb. 2, L. Mandel gave a talk at the Démons group, Univ. Paris 11, Orsay.
- Feb. 5, L. Mandel gave a talk at the Mimosa project, INRIA, Sophia.

- Feb. 21, L. Mandel gave a talk at the Sardes project, INRIA, Grnoble.
- Feb. 22, L. Mandel gave a talk at the Plume, ENS, Lyon.
- Mar. 12, L. Mandel gave a talk at the ALIDECS ACI, Paris.
- May. 31, L. Mandel gave a talk at the List group CEA, Saclay.
- Apr. 26, L. Mandel gave a talk at the CAPP group, LIG, Grenoble.
- September, P.-M. Deniélou and R. Corin participated in the Salon de la Recherche.

8.3.3. Visits

- Jan 21-25, Aquinas Hobor (Princeton) visited Rocquencourt for collaboration with F. Zappa Nardelli.
- Jun 12-Jul 20, Aquinas Hobor (Princeton) spent one month in the Moscova Project-Team for collaboration with F. Zappa Nardelli.
- Jun 18-Jul 17, Andrew Appel (Princeton) spent one month in the Moscova Project-Team for collaboration with F. Zappa Nardelli.
- Jul 9-Jul 10, Matthew Parkinson (Cambridge) visited Rocquencourt for collaboration with A. Appel, A. Hobor and F. Zappa Nardelli.
- May-Jun-Jul, Vidyut Ghosal and Arun Kumar Patala from IIT Kanpur spent three months with us to work on Burfiks under the direction of F. Zappa Nardelli. They were supported by the program “Internship at INRIA for international students”.

9. Bibliography

Major publications by the team in recent years

[1] 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.

[2] R. CORIN, P. M. DENIELOU, C. FOURNET, K. BHARGAVAN, 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/.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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://pauillac.inria.fr/~leifer/research.html.

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

[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.

Year Publications

Articles in refereed journals and book chapters

[10] T. BLANC, J.-J. LÉVY, L. MARANGET. *Sharing in the weak lambda-calculus revisited*, 2007.

[11] R. CORIN, P. M. DENIELOU, C. FOURNET, K. BHARGAVAN, J. LEIFER. *A Secure Compiler for Session Abstractions*, in "Journal of Computer Security", To appear, 2007.

[12] J.-J. LÉVY. *Algorithmique et programmation – Structures de Données*, ISBN:2-7117-4846-4, Vuibert, 2007.

[13] L. MANDEL, M. POUZET. *ReactiveML : un langage fonctionnel pour la programmation réactive*, in "Technique et Science Informatiques (TSI)", Accepted for publication, 2007.

[14] L. MARANGET. *Warnings for pattern matching*, in "Journal of Functional Programming", vol. 17, May 2007.

[15] 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.

Publications in Conferences and Workshops

[16] A. APPEL, S. BLAZY. *Separation Logic for Small-step C Minor*, in "20th Int. Conference on Theorem Proving in Higher Order Logics", 2007.

[17] A. W. APPEL, P.-A. MELLIES, C. D. RICHARDS, J. VOUILLON. *A Very Modal Model of a Modern, Major, General Type System*, in "34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Nice", January 2007.

[18] K. BHARGAVAN, R. CORIN, C. FOURNET. *Crypto-Verifying Protocol Implementations in ML*, in "Proceedings of the 3rd Workshop on Formal and Computational Cryptography (FCC'07), Venice, Italy", M. BACKES, Y. LAKHNECH (editors), To appear, July 2007, http://www.msr-inria.inria.fr/projects/sec/fs2cv/.

[19] R. CORIN. *Computational Soundness of Formal Encryption in Coq*, in "Proceedings of the 3rd Workshop on Formal and Computational Cryptography (FCC'07), Venice, Italy", M. BACKES, Y. LAKHNECH (editors), To appear, July 2007.

[20] R. CORIN, P. M. DENIELOU. *A Protocol Compiler for Secure Sessions in ML*, in "3rd Symposium on Trustworthy Global Computing (TGC'07), Sophia, France", To appear, LNCS, November 2007.

[21] R. CORIN, P. M. DENIELOU, C. FOURNET, K. BHARGAVAN, 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/.

[22] C. FOURNET, N. GUTS, F. ZAPPA NARDELLI. *A Formal Implementation of Value Commitment*, in "17th European Symposium on Programming (ESOP'08)", To appear, April 2008.

[23] A. HOBOR, A. APPEL, F. ZAPPA NARDELLI. *Oracle Semantics for Concurrent Separation Logic*, in "17th European Symposium on Programming (ESOP'08)", To appear, April 2008.

[24] L. MANDEL, L. MARANGET. *Programming in JoCaml*, in "European Symposium on Programming (ESOP'08)", To appear, April 2008.

[25] L. MOREL, L. MANDEL. *Executable Contracts for Incremental Prototypes of Embedded Systems*, in "Formal Foundations of Embedded Software and Component-Based Software Architectures (FESCA'07)", March 2007.

[26] S. SARKAR, P. SEWELL, F. ZAPPA NARDELLI. *Specifying real-world binding structures*, in "WMM 2007 – Informal ACM SIGPLAN Workshop on Mechanised Metatheory", 2007.

[27] P. SEWELL, F. ZAPPA NARDELLI, S. OWENS, G. PESKINE, T. RIDGE, S. SARKAR, R. STRNIŠA. *Ott: Effective Tool Support for the Working Semanticist*, in "Proceedings of ICFP 2007: the 12th ACM SIGPLAN International Conference on Functional Programming (Freiburg)", 12pp, 2007.

Internal Reports

[28] L. MANDEL, L. MARANGET. *Programming in JoCaml – Extended version*, Research report, n^{o} 6261, INRIA, 2007, http://hal.inria.fr/inria-00166125.

Miscellaneous

[29] J. ALGLAVE. *Un prouveur de théorèmes certifié pour la logique de séparation*, 2007.

[30] L. MARANGET, L. MANDEL, Q. MA. *JoCaml, release 3.09*, June 2007, http://jocaml.inria.fr/.

References in notes

[31] P. BAPTISTE, L. MARANGET. , École polytechnique, August 2006, http://www.enseignement.polytechnique.fr/profs/informatique/Luc.Maranget/421/poly.

[32] T. BLANC. *Propriétés de sécurité dans le lambda-calcul*, Ph. D. Thesis, Ecole Polytechnique, 2006.

[33] T. BLANC, J.-J. LÉVY, L. MARANGET. *Sharing in the weak lambda-calculus*, LNCS, n^{o} 3838, Springer, December 2005.

[34] D. F. C. BREWER, M. J. NASH. *The Chinese Wall Security Policy*, in "Proceedings of the 1989 IEEE Symposium on Security and Privacy", 1989, p. 206–214.

[35] C. FOURNET, A. D. GORDON. *Stack Inspection: Theory and Variants*, Technical report, n^o MSR–TR–2001–103, Microsoft Research, 2001.

[36] P. GRAHAM. *A plan for spam*, 2002, http://www.paulgraham.com/spam.html.

[37] T. JENSEN, D. L. MÉTAYER, T. THORN. *Verification of control flow based security propertie*, in "Proceedings of the 1999 IEEE Symposium on Security and Privacy", IEEE Computer Society Press, 1999, p. 89–103.

[38] X. LEROY. *Formal certification of a compiler back-end, or: programming a compiler with a proof assistant*, in "33rd symposium Principles of Programming Languages", ACM Press, 2006, p. 42–54.

[39] J.-J. LÉVY. *Réductions correctes et optimales dans le lambda-calcul*, Ph. D. Thesis, Université Paris 7, 1978.

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

[41] 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.

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

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

[44] J. REYNOLDS. *Separation logic: a logic for shared mutable data structures*, in "Invited paper, LICS'02", 2002.

[45] F. B. SCHNEIDER. *Enforceable security policies*, in "ACM Transactions on Information and System Security", vol. 3, n^o 1, February 2000, p. 30–50.

[46] V. SIMONET. *Inférence de flots d'information pour ML: formalisation et implantation*, Ph. D. Thesis, Université Paris 7 - Denis Diderot, 2004.

[47] 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.

[48] N. ÇAĞMAN, J. R. HINDLEY. *Combinatory Weak Reduction in Lambda Calculus*, in "Theoretical Computer Science", vol. 198, 1998, p. 239–249.