7. Plume - Proof Theory and Formal Semantics

Team: PLUME
Scientific leader: Olivier LAURENT
Evaluation 2006-2009

Web site: http://www.ens-lyon.fr/LIP/PLUME/
Parent Organisations: École Normale Supérieure de Lyon, Université of Lyon, CNRS
Team History: The Plume team has been started by Christine Paulin, with a main focus on machine-assisted proofs and the Coq system. After Pierre Lescanne took the lead of Plume in 1997, the themes of Plume have been extended to encompass programming semantics and proof theory in a broader sense. Daniel Hirschkoff then became scientific leader of the team in September 2006. In September 2008, the team has started a major evolution, with the arrival of three researchers (P. Baillot, O. Laurent, A. Miquel) and the definition of new research directions including linear logic, implicit computational complexity and program extraction from classical proofs.

7.1 Team Composition

Current team members

<table>
<thead>
<tr>
<th>Name</th>
<th>First name</th>
<th>Function</th>
<th>Institution</th>
<th>Arrival date</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDEBAUD</td>
<td>Philippe</td>
<td>Associate Professor (MdC)</td>
<td>ENSL</td>
<td>01/09/1992</td>
</tr>
<tr>
<td>BAILLOT</td>
<td>Patrick</td>
<td>Research Scientist (CR)</td>
<td>CNRS</td>
<td>01/09/2008</td>
</tr>
<tr>
<td>DUPRAT</td>
<td>Jean</td>
<td>Associate Professor (MdC)</td>
<td>ENSL</td>
<td>01/09/1987</td>
</tr>
<tr>
<td>HIRSCHKOFF</td>
<td>Daniel</td>
<td>Associate Professor (MdC)</td>
<td>ENSL</td>
<td>01/09/1999</td>
</tr>
<tr>
<td>LAURENT</td>
<td>Olivier</td>
<td>Research Scientist (CR)</td>
<td>CNRS</td>
<td>01/09/2008</td>
</tr>
<tr>
<td>LESCANNE</td>
<td>Pierre</td>
<td>Professor (Prof)</td>
<td>ENSL</td>
<td>01/09/1997</td>
</tr>
<tr>
<td>RIBA</td>
<td>Colin</td>
<td>Associate Professor (MdC)</td>
<td>ENSL</td>
<td>01/09/2009</td>
</tr>
</tbody>
</table>

Ph. Audebaud was at the University of Nice until Sept. 2006. C. Riba joined the team on an ATER position in Sept. 2008 (see next page).

Post-docs, engineers and visitors

<table>
<thead>
<tr>
<th>Name</th>
<th>First name</th>
<th>Function and % of time (if other than 100%)</th>
<th>Institution</th>
<th>Arrival date</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIQUEL</td>
<td>Alexandre</td>
<td>détachement</td>
<td>ENSL</td>
<td>01/09/2009</td>
</tr>
<tr>
<td>TRANQUILLI</td>
<td>Paolo</td>
<td>postdoc</td>
<td>ENSL</td>
<td>21/09/2009</td>
</tr>
<tr>
<td>VECCHIATO</td>
<td>Silvia</td>
<td>doctoral internship</td>
<td>Univ. Siena</td>
<td>01/09/2009</td>
</tr>
</tbody>
</table>

A. Miquel joined the team on a CNRS délégation position in Sept. 2008 (see next page).

Doctoral Students

<table>
<thead>
<tr>
<th>Name</th>
<th>First name</th>
<th>Institution</th>
<th>Date of first registration as doctoral student</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEMANGEON</td>
<td>Romain</td>
<td>ENSL</td>
<td>01/09/2007</td>
</tr>
<tr>
<td>LASSON</td>
<td>Marc</td>
<td>ENSL</td>
<td>01/09/2009</td>
</tr>
<tr>
<td>PARDON</td>
<td>Aurelien</td>
<td>ENSL</td>
<td>01/09/2006</td>
</tr>
<tr>
<td>PETIT</td>
<td>Barbara</td>
<td>ENSL</td>
<td>01/09/2008</td>
</tr>
</tbody>
</table>

Past team members

<table>
<thead>
<tr>
<th>Name</th>
<th>First name</th>
<th>Position</th>
<th>Parent Institution</th>
<th>Arrival date (or Oct. 2005)</th>
<th>Departure date</th>
<th>Current position</th>
</tr>
</thead>
</table>

Evolution of the team:

The return of Ph. Audebaud after a détachement at the University of Nice in Sept. 2006 has been followed by the departure of T. Hirschowitz to the University of Chambéry in Sept. 2007. In Sept. 2008, P. Baillot, O. Laurent and A. Miquel joined the team (which was then composed of 4 permanent members). This has had a major impact on the scientific goals of the Plume team, with a new strong research direction on proof theory. This evolution is continued with the recruitment of C. Riba on a maître de conférences position in Sept. 2009.

7.2 Executive summary

**Keywords** Semantics of programming languages, computer assisted proofs, proof theory.

**Research area** Researchers in the Plume team study methods for the formal analysis of computer programs and, more generally, of computing systems. More precisely, we work on the formal definition of (aspects of) programming languages, and of static analyses of programs. We also rely on theorem proving systems to develop rigorous formalizations and proofs about mathematical theories.

We build on the proofs-as-programs correspondence to strengthen the links between proof theory in logic and computer science. We deal in particular with the integration of additional programming features: concurrency, mobility, modularity, probabilistic behaviours, ...

**Main goals** The main goals of the team are:

- to strengthen the understanding of various programming constructs (control operators, probabilistic primitives, concurrency, cryptographic aspects);
- to provide logical foundations for modern programming languages;
- to build type systems controlling the behaviour of programs (termination, complexity bounds, dynamic modularity);
- to define extraction mechanisms in order to build certified software from formal proofs of specifications;
- to develop rigorous formalizations of mathematical theories with proof assistants like Coq.

**Methodology** Our tool-set comes from mathematical logic and the semantics of programming languages. We use the Curry-Howard *(i.e. proofs-as-programs)* correspondence as a bridge between proof-theory and computer science. In this way we are able to build mathematical reasonings about program behaviours.

Our publications often come with a mechanized verification (as Coq contributions) of the results they contain.

More recently, we have put a specific focus on the use of (variants of) linear logic and of the realizability theory as unifying tools to develop the foundations of programming languages.

**Highlights**
New up-to techniques for weak bisimulation. Reasoning formally about distributed and concurrent systems is a complex task, because it may lead to a blow up of the number of states to be taken into account, and can involve the analysis of rich and subtle behaviours. Up-to techniques are a set of tools that have been proposed to help establishing properties of equivalence between two concurrent systems (a distributed protocol and its optimized version, a specification of a server and its implementation, ...).

We have proposed a uniform and modular theory of up-to techniques for weak bisimulation that captures existing proof technology and allows us to define new non-trivial techniques based on termination arguments. This work has fostered the start of a new activity on the machine-assisted formalization of proofs about concurrent systems.

Classical program extraction in Coq. The classical extraction module for Coq enriches the Coq proof assistant with a set of commands to extract programs from classical proofs. The target language of the extraction mechanism is the $\lambda_c$-calculus, an extension of the $\lambda$-calculus with control primitives introduced by Jean-Louis Krivine to realize provable formulas of classical logic. This allows us to study the computational content of proofs formalized in the calculus of constructions with universes.

Expressiveness and separability of spatial logics for concurrency. Several modal logics have been proposed as specification formalisms to analyze the behaviour and the structure of concurrent processes. Among these, the Ambient Logic (AL) has been introduced to express properties of process mobility in the calculus of Mobile Ambients (MA), and as a basis for query languages on semistructured data. We have established expressiveness properties of AL by showing how it is possible to define logical formulas that capture important properties of MA processes, related to their behaviour (how processes communicate, how they move along computation) but also to their structure (name occurrences, persistence). Building on these results, we have been able to compare the equivalence induced by the logic with other relevant equivalences on processes, and to establish (un)decidability properties of the former.

Proofs of randomized algorithms. Randomized algorithms are widely used for finding efficiently approximated solutions to complex problems (for instance primality testing) and for obtaining good average behaviour. Proving properties of such algorithms requires subtle reasoning both on algorithmic and probabilistic aspects of programs. We have defined a new method for proving properties of randomized algorithms in a proof assistant based on higher-order logic. It is based on the monadic interpretation of randomized programs as probabilistic distributions (Giry, Ramsey and Pfeffer).

Termination and concurrency. Termination is a fundamental property in sequential languages but it is also important in concurrency. For instance, termination can be used to guarantee that interaction with a resource will eventually end (avoiding denial of service situations), or to ensure that the participants in a transaction will eventually reach an agreement. Termination is particularly difficult in the $\pi$-calculus (a commonly accepted model for concurrent and mobile computation), due to the expressiveness of this formalism. We have introduced type systems for termination based on term-rewriting techniques. Both the $\pi$-calculus (with name passing) and its higher-order extension with process passing are studied.

7.3 Research activities

In order to guarantee safety and security, programs should be developed in a controlled way, and appropriate methods should be devised. Numerous formal methods have been introduced to help programmers in pursuing this goal. Depending on the application domain, the appropriate notion of being controlled may strongly differ (e.g. termination vs. liveness). The spectrum of properties that such methods make possible to address includes: security, certification, termination, resource boundedness, deadlock freedom, liveness, control on information flow, ... Inside the large body of work on formal methods, we can distinguish between static approaches (systems are analysed before execution) and dynamic approaches (analysis is done along execution). We focus on static approaches, and more precisely type systems and formal semantics, using logical means.

Researchers in the Plume team study methods for the formal analysis of computer programs and, more generally, of computing systems. More precisely, we work on the formal definition of (aspects of) programming languages, and of static analyses of programs. We also rely on theorem proving systems to develop rigorous formalizations and proofs about mathematical theories. We build on the proofs-as-programs (Curry-Howard) correspondence to strengthen the links between proof theory in logic and computer science. The Curry-Howard correspondence which relates proofs and programs provides us with a bridge between proof-theory and computer science. It is a very powerful framework for technology transfers between these two research topics. In particular formal proofs are endowed with a computational meaning (cut-elimination is evaluation) and logic gives a crucial tool to analyze the foundations of programming languages. The meaning of this correspondence is very well understood in the intuitionistic/functional case, which is the original setting. It has been extended to classical logic and control operators (like call/cc) in the last twenty years. We are now looking for other extensions to develop logical foundations of additional programming concepts: concurrency, mobility, complexity control, ... On the programming languages side, the team studies the formal description of expressive programming constructs (such as probabilistic operations, primitives for modularity, or concurrency mechanisms). This is done by following the approach of formal semantics of programming features. It concerns both operational semantics: the precise definition of the computational rules underlying a programming primitive, and denotational semantics: the search for invariants of computation. Our aim is to define and analyze the properties of formalisms that capture these programming constructs in a concise and clear fashion, in the same way the $\lambda$-calculus can be used to
model sequential functional computation. In doing so, we mostly rely on mathematical tools, to develop the operational and denotational semantics of programs (partial orders, category theory, ...). We often use the obtained models to design type systems allowing to characterize and to analyze specific behavioural properties of programs. Due to the very formal nature of the objects we manipulate, we are frequently turning our results into developments in the Coq proof assistant. In a more general way, we also use Coq for the formalization of various mathematical theories.

7.3.1 Curry-Howard correspondence

We work on various aspects of the Curry-Howard correspondence: from the traditional (twenty years old) question of classical logic to more recent points such as the implicit complexity point of view. Consequences in computer science are obtained by means of realizability and program extraction from proofs.

Computational interpretations of classical logic

List of participants O. Laurent (CR CNRS), P. Lescanne (PR ENS Lyon), D. Zunic (PhD 2004-2007)

Keywords Sequent calculus, graphical syntaxes for proof-theory, constructive classical logic

Scientific issues, goals, and positioning of the team The Curry-Howard correspondence, originally introduced between intuitionistic logic and functional languages, is now well understood in the more general context of classical logic and control operators. This led to the development of many logical systems for presenting proofs in classical logic with a well defined computational meaning. The Plume team took part in this line of work and now focuses more on clarifying the relations between the various proposals given in the literature.

Major results Plume members worked in particular on the use of graphical syntaxes for representing proofs from classical logic. The \( \lambda \) diagrammatic calculus allows for a graphical analysis of the sequent calculus of classical logic. This belongs to the family of classical systems with non-deterministic computational behaviours. On the other side, id-nets give a deterministic representation of classical proofs. These nets are used to provide us with a unified framework for interpreting the deterministic classical systems. This is based on a strong relation with intuitionistic logic and works both for call-by-name and for call-by-value classical systems.

Self-assessment The theory of the deterministic computational behaviour of classical logic is now well understood and appropriately unified. On the other hand there are still various proposals for non-deterministic systems. In the same way the relation between the deterministic and the non-deterministic cases are still not completely well analyzed.

Program extraction from proofs

List of participants Ph. Audebaud (MdC ENS Lyon), L. Chiarabini (visitor 2007), A. Miquel (CNRS Delegation 2008-2009)

Keywords proof assistant, program extraction, control operators

Scientific issues, goals, and positioning of the team Not only proofs are programs, but if \( \pi \) is a proof of the formula \( F \), the program \( P \) corresponding to \( \pi \) satisfies the specification formalized by \( F \). This provides us with a way of developing certified software without the usual two stages approach (first write the code and then prove the correctness with respect to the specification). After a formalization of the specification of the wanted behaviour of the program through a logical formula, the developer interacts with a proof assistant to write down a proof of the formula. The code is then automatically extracted from the proof and guaranteed to satisfy the specification.

Major results Efficient program extraction from proofs can benefit from various backgrounds. Ph. Audebaud and L. Chiarabini approach aims at applying well known techniques from the programming community to the proof side; the current work concerns tail recursion and defunctionalization. Following Krivine’s work on realizability for classical logic, A. Miquel has shown how to define realizability interpretations for the calculus of inductive constructions (the type theory underlying Coq). Starting from this theoretical result, he developed an extraction module for the Coq proof assistant, addressing concrete issues related with the representation of integers for example.

Self-assessment The abstract theory of realizability is now mature for transfer towards software development. This is the line we are focusing on. The practical use of the theory will naturally bring new questions on the more abstract side.

Implicit Computational Complexity

List of participants P. Baillot (CR CNRS)

Keywords Light linear logics, type systems for complexity

Scientific issues, goals, and positioning of the team The linear logic (LL) approach to implicit computational complexity aims at defining variants (or restrictions) of LL for which the cut-elimination procedure encodes precisely a given complexity class. In this way, the complexity of programs becomes an additional computational behaviour accessible through logical means. We put a particular focus on two specific questions: the semantic analysis of the derived logical systems and the design of associated type systems.

This is one of the newly started research directions of the team (with the arrivals of P. Baillot and O. Laurent).
Major results P. Baillot and D. Mazza (Univ. Paris 13) have proposed a variant of linear logic called linear logic by levels (L3), which admits an elementary time complexity bound on proof normalization, and at the same time generalizes and simplifies the system ELL of Girard. A subsystem of this linear logic by levels also corresponds to polynomial time computation and generalizes the system LLL (light linear logic). One benefit of these systems with respect to ELL and LLL is that they have simpler definitions with proof-nets (the graphic representation of proofs).

Self-assessment The back and forth methodology between linear logic and type systems has been very successful in the elaboration of L3. It goes in the important general direction of weakening the constraints in light linear logics in order to be able to deal with more algorithms.

Realizability and higher-order term rewriting


Keywords Realizability, reducibility candidates, termination

Scientific issues, goals, and positioning of the team We have already seen how realizability refines typing and helps in the computational interpretation of proofs. More generally it is a key tool in the study of deep properties of computation, such as termination. Strong progress has been made on the understanding of the relations between reducibility, realizability and orthogonality. It helps in the development of applications to various extensions of the $\lambda$-calculus.

Major results Union types are required for typing various extensions of the $\lambda$-calculus but are known to interact sometimes badly with preservation under reduction, termination and realizability techniques in general. Through the introduction of a notion of value inspired by reducibility candidates, C. Riba has designed typing rules for union types which are preserved under reduction, ensure termination, ... This approach applies also to implicit existential types, since they are interpreted as infinitary unions in realizability.

The $\lambda$-calculus with constructors is designed for the abstract study of pattern matching mechanisms. In order to control the dynamics of this very rich calculus, B. Petit has used a type system based on system F and subtyping whose correction is derived through a realizability model.

Self-assessment The Plume team now has a strong expertise in realizability techniques both on the fundamental aspects, on the relation with logical systems and on applications to term rewriting. This has helped a lot in the recent results. A deeper look at relations with denotational semantics is something to address.

7.3.2 Structures of programming languages

We do not only rely on logic to study programs and in a more general way we are interested in mathematical tools for understanding various structures appearing in computing. This line of research is related to the previous one (logic and proof theory), while being closer to programming languages, for which we try to build (semantic) models.

Modular presentations of programming languages

List of participants T. Hirschowitz (CR CNRS 2004-2007), A. Pardon (PhD 2006-???)

Keywords syntax and operational semantics of programming languages, category theory

Scientific issues, goals, and positioning of the team We work on enhancing the available tools to present and analyze the syntax and operational semantics of calculi that are used to describe various forms of computation and interaction. These calculi include the $\lambda$-calculus, the $\pi$-calculus, as well as Milner et al.’s more recent generalization of a vast amount of formalisms, known as bigraphs.

We rely for this on a category-theoretic approach, the main goal being to improve the modularity of presentation, and to clarify the use of binders in such formalisms.

Major results We have been studying the use of symmetric monoidal closed theories to represent binding in (formal models of) programming languages. This provides a modular notion of syntax, which has allowed us to revisit the notion of context in languages with binders. We have thus been able to accommodate the $\lambda$-calculus and the $\pi$-calculus in our setting, and to propose an extended version of bigraphs.

Self-assessment In order to extend the setting we have introduced and to be able to define the operational semantics of programs, we would like to reformulate our contribution in the framework of double categories, along the lines of previous work by Melliès on models for Multiplicative Linear Logic.

Certification of randomized algorithms

List of participants Ph. Audebaud (MdC ENS Lyon)

Keywords randomized algorithms, formal proof, denotational semantics
Scientific issues, goals, and positioning of the team This research takes place as part of the ANR project Scalp which aims at providing formal models for presenting and certifying cryptographic protocols and algorithms.

More specifically, Ph. Audebaud works with Christine Paulin (LRI, Orsay) on developing a well fitted higher-order functional framework.

Major results We have designed a programming language which meets requirements with respect to expectations in the area of cryptography. It is currently delegated under implementation. For this, the Why verification framework has been chosen, as it already provides most of the required technology, and because a reasonable amount of modification should be needed to reach our goal. We work in parallel on enhancing the proof assistants technology (specifically, the Coq system) with an efficient and easy to use library in order to help users in developing formal proofs. We take advantage of recent tools such as Class Types (from Haskell) at least for internal representation of the objects involved.

We currently address the question of semantics with the help of a different language, the calculus $\lambda_c$, designed by Thurn, Pfenning and Park. This language is more explicit as far as randomized effects are concerned, which helps at a meta level.

Self-assessment A central aspect of this project is the close integration between its various research directions, that cover the spectrum ranging from abstract denotational models to core programming languages, and to machine mechanization of formal proofs.

Concurrency Theory

List of participants R. Demangeon (PhD 2007-??), D. Hirschkoff (MdC ENS Lyon), D. Pous (PhD 2005-2008)

Keywords concurrency, process calculus, operational equivalence, type system

Scientific issues, goals, and positioning of the team Process calculi such as CCS or the $\pi$-calculus provide a framework where it is possible to represent many aspects of concurrent and distributed programming: message passing, dynamic reconfiguration of communications links, localized (and distant) communications, code mobility, persistent servers, ... In this setting, a program is given by a process. We develop methods for the analysis of the behaviour of processes: guarantee statically some properties at runtime, compare the behaviours of processes, compare different calculi in terms of the behaviours they can express.

Major results We have deepened the understanding of operational equivalences, which give rise to the notion of behaviour of processes. Behaviours provide a form of denotation, beyond the mere syntax of terms. We have studied algebraic properties and axiomatisations of behavioural equivalences, in relation with expressiveness of syntactic operators. We have also worked on proof techniques for behavioural equivalences, and studied in particular how one can combine several techniques to obtain more powerful proof tools.

Another research direction is concerned with type systems for termination in concurrent system. We have introduced such static analyses for the first- and higher-order cases (name and process passing, respectively).

Self-assessment The effort towards a deeper and more unified understanding of techniques for behavioural equivalences has led to a satisfying level of maturity. A book chapter about up-to techniques for concurrency is currently being written, and a new research activity on the mechanisation of reasoning about concurrent systems in a theorem prover has been started.

The work on termination should help in developing bridges from the theory of sequential (functional) computation, as expressed in the $\lambda$-calculus, and the world of concurrency. It seems particularly relevant for this to be able to adapt powerful $\lambda$-calculus techniques to the $\pi$-calculus, as we have started to do.

Dynamic Modularity


Keywords process calculus, distributed programming, abstract machine, type system

Scientific issues, goals, and positioning of the team We collaborate with the Sardes project (INRIA Rhône Alpes) on the design of process calculi-based models for dynamic modularity. This phrase stands for typical aspects of component-based programming, such as code mobility, dynamic update of modules, and reflexive programming. We want to provide a clean, mathematically defined formalism where forms of dynamic modularity can be expressed and analyzed. This formalism can serve as the basis for the design of new programming primitives.

This work has been supported by the ANR project “MoDyFiable - Modularité Dynamique Fiable” (2005-2008).

Major results We have developed a formalism where it is possible to express some forms of dynamic modularity. This process calculus, called $k\pi$, features a primitive for passivation, which provides the ability to freeze a running process in order to manipulate it (send it to a different location, duplicate it, replace it with a newer version, modify some of its code). We have defined an abstract machine for $k\pi$, which demonstrates how passivation can be used in a distributed setting. A prototype implementation of this machine has raised interesting questions related to the design of appropriate type systems in order to guarantee forms of runtime safety for $k\pi$ processes.

Self-assessment The primitive of passivation is very expressive, and goes beyond previously existing forms of interaction in process calculi (notably higher-order communication and forms of code mobility). Several difficult questions related to the metatheory of calculi featuring passivation deserve to be studied. Alternatively, one could also be interested in finding out whether it is possible to define controlled forms of passivation, that renounce to some of its expressiveness in favor of more tractable theoretical properties.
7.3.3 Formalizations in the Coq proof assistant

The formalization activities of the team address the machine representation of different mathematical theories through the development of computer assisted proofs (mainly with the Coq proof assistant).

Game theory

**List of participants**  P. Lescanne (PR ENS Lyon), S. Le Roux (PhD 2004-2008)

**Keywords**  Nash equilibria, induction, co-induction

**Scientific issues, goals, and positioning of the team**  Various concepts in game theory are derived in a not formal enough way. P. Lescanne and S. Le Roux defined a notion of Feasibility-Desirability games (FD-games) for unifying the view of decision theory, strategic games and evolutionary games. This comes with the study of Nash equilibria for such games.

**Major results**  A model of coalition games based on FD-games has been developed with F. Delaplace with applications in modelling gene regulation activities.

P. Lescanne revisited infinite sequential games using co-induction, showing that previous analyses of those games were inadequate, especially concerning "escalation" which happens to be a Nash equilibrium in the co-inductive formalization.

**Self-assessment**  Formal approaches to game theories allow us to identify under-specified notions appearing in the literature. The introduction of logical notions such as co-induction helps in clarifying the theory by proving that behaviour usually supposed to be irrational can be shown rational in the appropriate framework.

Constructive geometry

**List of participants**  J. Duprat (MdC ENS Lyon)

**Keywords**  Euclidean geometry, constructive mathematics, axiomatics

**Scientific issues, goals, and positioning of the team**  Starting from strong similarities between constructive proofs and proofs in Euclidean geometry, J. Duprat studies axiomatizations of geometry in the Coq proof assistant. The objective is to provide an interactive framework for describing geometric pictures and then formally proving, in the Coq proof assistant, the properties suggested and induced by the picture.

**Major results**  Based on two primitive notions (orientation and equi-distance) and three constructions (lines, circles and intersection points), J. Duprat defined an axiomatization of constructive points of the plane. Current developments use constructive points of a given line to define distances and constructive points of the unit circle to define angles.

**Self-assessment**  The interaction between graphical tools and a proof assistant is very promising for providing a framework for formal proofs in geometry. The current axiomatization allows one to derive all the Hilbert axioms (except of course the continuity axiom).

7.4 Application domains and social or economic impact

As explained above, programs represent our main target: we study methods to mathematically define and analyze the behaviour of programs. However, we are not addressing the implementation of actual tools to check properties of real size programs, but work on the extension of the mathematical underpinnings of logical methods for the verification of software and the development of certified software.

7.5 Visibility and Attractivity

7.5.1 Prizes and Awards

**Awarded communications:**
- Best student paper award, Damien Pous, ICALP conference, 2005
- Best student paper award, Barbara Petit, TLCA conference, 2009

**Chairing and organization of conferences:**
- *Mathematics of Program Construction* (MPC’08), Philippe Audebaud, co-chair.

The colloquium *A journey through term rewriting and lambda-calculi* has been organized at Nancy in 2008 for the 60th birthday of Pierre Lescanne.

7.5.2 Contribution to the Scientific Community

**Management of Scientific Organisations**

Pierre Lescanne has been president of SPECIF (*Société des Personnels Enseignants et Chercheurs en Informatique de France*) between 2006 and 2008 (and vice-president for international questions in 2008–2009).
Administration of Professional Societies

Editorial Boards

- Applicable Algebra in Engineering, Communication and Computing (Springer), Pierre Lescanne (member of the editorial board).
- Linear Logic wiki (http://llwiki.ens-lyon.fr/), Olivier Laurent (coordinator).

Organisation of Conferences and Workshops

- Mathematics of Program Construction (MPC’08), Philippe Audebaud
- Journées GEOCAL-LAC 2009 du GDR IM, Pierre Lescanne and Alexandre Miquel
- Computational Logic and Applications (CLA 2007) Cracovie, Pierre Lescanne
- Computational Logic and Applications (CLA 2008) Cracovie, Pierre Lescanne
- Computational Logic and Applications (CLA 2009) Lyon, Pierre Lescanne
- Congrès SPECIF Saint Etienne 2007, Pierre Lescanne
- Congrès SPECIF Bordeaux 2008, Pierre Lescanne

Program committee members

- Concurrency Theory (CONCUR’07), Daniel Hirschkoff, PC member, 2007
- International Colloquium on Automata, Languages and Programming (ICALP’07), Pierre Lescanne, PC member, 2007
- International Conference on Typed Lambda Calculi and Applications (TLCA’09), Patrick Baillot, PC member, 2009
- Mathematics of Program Construction (MPC), Philippe Audebaud, MPC ’08 co-chair and MPC’10 PC member

International expertise

National expertise

- Selection committees (Maître de conférences, Professeur and Chargé de Recherche positions)
  - Université de Savoie, 2009, Pierre Lescanne (professor position)
  - Université d’Évry, 2009, Pierre Lescanne
  - Université Aix-Marseille II, 2009, Olivier Laurent
  - Université Aix-Marseille II, 2009, Alexandre Miquel
  - ENS Lyon, 2009, Olivier Laurent
  - ENS Lyon, 2009, Daniel Hirschkoff
  - Université Bordeaux I, 2009, Daniel Hirschkoff
  - INRIA Saclay, 2009, Daniel Hirschkoff (CR1 and CR2 positions)
  - ENSIMAG, 2006-2008, Pierre Lescanne
  - ENS Lyon, 2006-2008, Pierre Lescanne
  - Université Paris 7, 2006 and 2007, Daniel Hirschkoff
  - Université Joseph Fourier, Grenoble, 2007, Tom Hirschowitz
  - Université de Provence, 2006, Daniel Hirschkoff
- PhD committees
  - Matthieu Manceny, Université d’Évry, P. Lescanne, Nov. 2006
  - Jittisak Senachak, Japan Advanced Institute of Science and Technology, P. Lescanne, Jun. 2007
  - Sébastien Briais, EPFL, Lausanne (Switzerland), D. Hirschkoff, Dec. 2007.
  - Vincent Atassi, Université Paris 13, P. Baillot (co-adviser of the thesis), Dec. 2008
  - Jayshan Ragunandan, Imperial College, P. Lescanne, Jan. 2009
  - Guillaume Burel, Université Nancy 1, A. Miquel, Mar. 2009.
  - Romain Beauxis, École Polytechnique, P. Lescanne, May 2009
§7.6 Contracts and grants

7.6.1 External contracts and grants (Industry, European, National)

MoDyFiable Modularité Dynamique Fiable, ANR ARASSIA, 2006-2008. The partner is the Sardes project (INRIA Rhône Alpes). The scientific leader of the project is Tom Hirschowitz (member of Plume during the project). The goal of the project is to design the core of a programming language for dynamic modularity (distributed and mobile computation, runtime update of modules, inspection of running code), by exploiting process calculi-based techniques.

Galapagos Géométrie, algorithmes et preuves, ANR programme non-thématique (Blanc), 2007-2010. Project partners are INRIA (Sophia Antipolis), LSIT - UMR 7005 CNRS-ULP (Strasbourg), UFR Sciences, SP2MI (Poitiers). Web page: http://galapagos.gforge.inria.fr/. Project leader: Yves Bertot (INRIA). In this project, we study the applicability of theorem proving tools to two aspects of geometry. First we apply theorem proving tools to computational geometry. Second, we apply theorem proving tools to the verification of geometrical reasoning.

CHoCo Curry-Howard and Concurrency, ANR programme non-thématique (Blanc), 2008-2010. Project partners are the PPS laboratory (Univ. Paris 7) and the IML (Univ. Aix-Marseille 2). Web page: http://choco.pps.jussieu.fr/. The site leader for Lyon is Daniel Hirschkoff. The goal of the project is to work on the extension of the Curry-Howard correspondence to encompass concurrent phenomena, drawing inspiration from and making connection with concurrency theory.


CompIICE Complexité Implicite, Concurrence et Extraction, ANR programme non-thématique (Blanc), 1/1/2009- 31/12/2012. Project partners are ENS Lyon (coordinating site), Université Paris 13, INPL Nancy. Web page: http://www-lipn.univ-paris13.fr/complice/. Project leader: Patrick Baillot. Participants at LIP: Patrick Baillot, Romain Demangeon, Daniel Hirschkoff, Olivier Laurent. This project investigates implicit computational complexity, that is to say methods to statically ensure time and space complexity bounds on programs, and to characterize in this way complexity classes in a machine-independent way. It explores this problematic in the areas of functional programming, program extraction from formal proofs and concurrent systems.

Geocal Géométrie du Calcul, ACI Nouvelles Interfaces des Mathématiques, 2004-2006. Project partners are the PPS laboratory (Univ. Paris 7), the IML (Univ. Aix-Marseille 2), the LIPN (Univ. Paris 13). The site leader for Lyon is Daniel Hirschkoff. The goal of the project is to study proof theory and models of computation, developing a geometrical view of various computing and logical paradigms.

TLIT Types and Logic in Information Technologies, cooperation between CNRS and Serbian Ministry of Scientific Research, project leader are Silvia Ghilezan (University of Novi Sad, Serbia) and Pierre Lescanne (ENS de Lyon). The project which includes also among others Serbian Academy of Science, PPS (Paris), IRIT (Toulouse) aims at setting the logical basis of information theory using interpretation of classical logic, category theories and types in global computing.

Casimir Recherche coopérative en logique computationnelle, program Mira de la region Rhône-Alpes, project leader Pierre Lescanne (ENS de Lyon), between University Jagellone of Krakow, University of Savoie at Chambéry, University Joseph Fourier at Grenoble. The project investigates several aspects of computational logic including, among others, probabilistic aspects of lambda-calculus, a somewhat success story for the cooperation. Application of game theory to the study of modeling, program call for ideas 2007 of National Network on Complex System (RNSC) in coordination with Paris Institute of Complex Systems. The project was done in collaboration with Franck Delaplace from IBISC and University of Evry. Project leaders: Franck Delaplace and Pierre Lescanne.

7.6.2 Research Networks (European, National, Regional, Local)

D. Hirschkoff is a member of the BACON “équipe associée INRIA” between the Sardes project (INRIA Rhône Alpes) and Università di Bologna, for the year 2009.

The Plume team takes part in the “Geocal (géométrie du calcul)” and “LAC (logique, algèbre et calcul)” groups of the GDR “Informatique Mathématique”. It is also involved in the “Langages, Types et Preuves (LTP)” group of the GDR “Génie de la Programmation et du Logiciel”.

Monthly meetings of the CHoCo ANR project take place in Lyon and gather around 25 participants. http://choco.pps.jussieu.fr/events/.

A weekly seminar is organized by the Plume team in collaboration with the LIMD team of the LAMA laboratory (Chambéry). http://perso.ens-lyon.fr/tom.hirschowitz/GDT/.
7.6.3 Internal Funding

7.7 International collaborations resulting in joint publications

With grants

Università di Bologna, Italy (D. Sangiorgi)  D. Hirschkoff has a long standing collaboration with Davide Sangiorgi. This is testified by numerous joint publications [605, 633, 609, 611, 622, 615, 623, 624]. A joint PhD supervision (R. Demangeon) has started in Sept. 2007. A grant from the Université Franco-Italienne supports this PhD. More generally, support for the collaboration is provided by the ANR project “CHOCO”, as well as by the funding of an “équipe associée INRIA” between Bologna and the Sardes project at INRIA Rhône Alpes, to which D. Hirschkoff is attached. Their collaboration aims at studying various formal aspects of the $\pi$-calculus.

University of Novi Sad, Serbia (S. Ghilezan)  Pierre Lescanne has close contacts with Novi Sad, in particular through the TLIT project (see above). This includes the supervision of two Serbian PhD students in the past years (S. Likavec and D. Zunic). Their main research subject is the study of extensions of the $\lambda$-calculus to classical logic (such as Curien-Herbelin’s calculus in particular). They address strong normalization questions [626, 607] in collaboration with D. Dougherty (see below).

With joint publications

Worcester Polytechnic Institute, USA (D. Dougherty)  Pierre Lescanne and Dan Dougherty collaborate on various questions related to term rewriting with often a particular focus on the $\lambda$-calculus. Their work covers new term rewriting systems for describing object-based languages [625, 608] (together with L. Liquori) but also termination and classical extensions of the $\lambda$-calculus [626, 607].

Imperial College, UK (S. van Bakel)  Pierre Lescanne and Steffen van Bakel have developed a graphical formalism for continuation style computation, the $\mathcal{X}$ framework [644, 618]. Its expressive power is demonstrated by embedding various known systems such as the $\lambda$-calculus, Bloo and Rose’s calculus of explicit substitutions $\lambda x$, Parigot’s $\lambda \mu$, Curien and Herbelin’s calculus. It can also be seen as the pure untyped computational content of the reduction system for implicative classical sequent calculus of Urban.

JAIST, Japan (R. Vestergaard)  A collaboration between Pierre Lescanne and René Vestergaard aims at a better understanding of formal game theory. Together with Franck Delaplace (Genopole, Evry), they address biology related questions and they study how game theory allows us to study gene regulation networks [620].

7.8 Software production and Research Infrastructure

7.8.1 Software Descriptions

Up-to Techniques for Weak Bisimulation in Coq

• Type: theorem prover formalization
• Description.
This development is a formalization of a series of works on the theory of weak bisimilarity, and of up-to techniques for weak bisimulation (including several new up-to techniques). In particular, it includes formally checked proofs for a great part of [616, 617].

• URL: http://sardes.inrialpes.fr/~pous/upto/

Axiomatizations of Geometry in Coq

• Type: theorem prover formalization
• Description.
This library contains an axiomatization of the ruler and compass Euclidean geometry. Files A1 to A6 contain the axioms and the basic constructions. The other files build the proof that this axiomatization induces the whole plane geometry except the continuity axiom. For that the proofs of the Hilbert’s axioms conclude this work in the files E1 to E5.

• URL: http://pauillac.inria.fr/coq/contribs/RulerCompassGeometry.html

Distributed Abstract Machine

• Type: software.
• Description.
This development is a prototype implementation of the GCPan, an abstract machine for distributed and mobile computation introduced in [611]. It consists of an abstraction layer, developed as a generic OCaml library for message-passing (both in local and physically distributed modes), on top of which the actual abstract machine is defined.

• URL: http://sardes.inrialpes.fr/~pous/gcpan/
**Classical Extraction Module for Coq**

- **Type:** software.
- **Description.**

  The classical extraction module for Coq is an external module that enriches Coq with new commands to extract programs from classical proofs. It consists of two components:
  - A dynamically loadable module (**kextraction**), written in Caml and distributed under the LGPL license
  - A stand-alone evaluator (**jivaro**) to execute extracted programs, written in Caml and distributed under the GPL license

It is the first implementation of classical program extraction using the theory of classical realizability developed by Jean-Louis Krivine in the 90’s (and adapted to the Coq proof assistant). Extracted programs are expressed in the language \( \lambda_c \), a lazy functional language with continuations that significantly differs from the mainstream functional programming languages such as SML, Caml or Haskell, hence the need of a specific evaluator to execute extracted programs.

- **URL:** [http://perso.ens-lyon.fr/alexandre.miquel/kextraction/](http://perso.ens-lyon.fr/alexandre.miquel/kextraction/)

### 7.8.2 Contribution to Research Infrastructures

None.

### 7.9 Industrialization, patents, and technology transfer

None.

### 7.10 Educational Activities

#### 7.10.1 Supervision of Educational Programs

D. Hirschkoff is responsible for the first year of the Master in Theoretical Computer Science at ENS Lyon (2006-2009).

P. Baillot is in charge of the track “Informatique Mathématique” of the Master in Theoretical Computer Science at ENS Lyon, since May 2009.

Ph. Audebaud, J. Duprat, D. Hirschkoff and P. Lescanne regularly take part in the juries for the training periods of L3, M1 and M2 students at ENS Lyon.

J. Duprat participates to the second concours d’entrée at ENS Lyon in 2009.

#### 7.10.2 Teaching

<table>
<thead>
<tr>
<th>Name</th>
<th>Course title (short)</th>
<th>Level</th>
<th>Institution</th>
<th>No of hours</th>
<th>Number of academic years</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ph. Audebaud</td>
<td>Préparation</td>
<td>Agrég. maths</td>
<td>ENS Lyon</td>
<td>64</td>
<td>3</td>
</tr>
<tr>
<td>Ph. Audebaud</td>
<td>Prog2 (TD)</td>
<td>L3</td>
<td>ENS Lyon</td>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td>Ph. Audebaud</td>
<td>Prog2</td>
<td>L3</td>
<td>ENS Lyon</td>
<td>48</td>
<td>2</td>
</tr>
<tr>
<td>P. Baillot</td>
<td>Linear logic</td>
<td>M2</td>
<td>Paris 7</td>
<td>18</td>
<td>1</td>
</tr>
<tr>
<td>S. Briais</td>
<td>Prog. project</td>
<td>L3</td>
<td>ENS Lyon</td>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>J. Duprat</td>
<td>The Coq System</td>
<td>M</td>
<td>ENS Lyon</td>
<td>48</td>
<td>5</td>
</tr>
<tr>
<td>J. Duprat</td>
<td>Préparation</td>
<td>Agrég. maths</td>
<td>ENS Lyon</td>
<td>64</td>
<td>4</td>
</tr>
<tr>
<td>D. Hirschkoff</td>
<td>Prog1</td>
<td>L3</td>
<td>ENS Lyon</td>
<td>48</td>
<td>4</td>
</tr>
<tr>
<td>D. Hirschkoff</td>
<td>Concurrency Th.</td>
<td>M</td>
<td>ENS Lyon</td>
<td>48</td>
<td>2</td>
</tr>
<tr>
<td>D. Hirschkoff</td>
<td>Prog. project</td>
<td>L3</td>
<td>ENS Lyon</td>
<td>48</td>
<td>2</td>
</tr>
<tr>
<td>P. Lescanne</td>
<td>Prog2</td>
<td>L3</td>
<td>ENS Lyon</td>
<td>48</td>
<td>2</td>
</tr>
<tr>
<td>P. Lescanne</td>
<td>Prog2 (TD)</td>
<td>L3</td>
<td>ENS Lyon</td>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td>P. Lescanne</td>
<td>Th. Jeux</td>
<td>M</td>
<td>ENS Lyon</td>
<td>48</td>
<td>2</td>
</tr>
<tr>
<td>A. Miquel</td>
<td>Th. Demonstr.</td>
<td>M</td>
<td>ENS Lyon</td>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>C. Riba</td>
<td>Compil. project</td>
<td>L3 (TDs)</td>
<td>ENS Lyon</td>
<td>32</td>
<td>1</td>
</tr>
</tbody>
</table>

### 7.11 Self-Assessment

Since September 2008, the Plume team entered a transitional period towards the development of additional research directions, around logical foundations of programming languages. The next four years will mainly be dedicated to the establishment of these new activities and to the stabilization of our group.

**Strong points** The dynamism of the team has recently been renewed with the arrival of several young bright researchers to foster new and challenging research directions. Our team is highly attractive (as shown, in particular, by the number and quality of applications to permanent, post-doc, ATER,... positions).
We start to gather our expertises in the various aspects of the Curry-Howard correspondence to develop new interactions: complexity for concurrency, realizability techniques in implicit complexity, linear logic approaches to concurrency, semantics of quantification, game based interpretations of logic, ...

The current funding of the team is mainly provided by ANR projects and gives appropriate support for missions and invitations. In the specific environment of the ENS Lyon, the team plays an important role in the community through its investment in teaching. Among the French PhD students in our topics, an important proportion are former students of ENS Lyon.

**Weak points** After the retirement of Pierre Lescanne planned for 2012, there will be no senior (i.e. rank A) permanent position at Plume. With at least 5 junior (i.e. rank B) positions, the team will need (at least) two professors (or research directors) to deal appropriately with the team management.

The number of PhD students in the team is rather low. The arrival of new members and the increase of the number of “habilités à diriger des recherches” should have a strong impact on this point in the next years. However it should be noticed that it remains very difficult for good PhD students in our subjects to find permanent positions.

In the future we should probably try to add more diversity in our funding resources, in particular using bilateral funding and European funding.

**Opportunities** Initiated with the GeoCal ACI NIM (2004–2006), the Paris-Lyon-Marseille-Chambéry network plays a crucial role in the development of the research synergies at a national level around linear logic and the semantics of programming languages. This community is currently incarnated within the GDR Informatique Mathématique (in the GeoCal and LAC working groups in particular), and is at the same time probably too big for the model defined by ANR projects.

A by-product of this research dynamics is a number of good students who just finished their PhD or are about to defend it. This guarantees a very high level in the future recruitments.

The team could benefit from a better integration in the research environment in logic at Lyon, in particular through closer interactions with the mathematical logic team of University Lyon 1 (at Institut Camille Jordan), that mainly focuses on model theory. Some collaboration already started through the European MALOA (MAthematical LOgic and Applications) project for doctoral studies (kick-off planned in autumn 2009).

Strong relations with logic teams and people working in mathematical computer science in the area could be a good starting point for the development of master studies around these topics at Lyon.

Outside France, our team is involved in very strong international collaborations: Italy (Bologna, Roma, Torino, ...), UK (Cambridge, London, ...), Japan (Kyoto, JAIST, Nagoya), USA (Worcester, Boston, ...), Serbia (Novi Sad), Poland (Krakow), Germany (München), ...

**Risks**

Dealing with the growth of a research team, in times where important changes impacts the French academic life, entails an increasing amount of time spent on administrative duties, not for research.

In the specific case of funding, dealing with short term research projects compels us to do frequent re-applications (which are very time-demanding). In lack of sufficient recurrent funding, the resources of the team are currently only guaranteed for the next two or three years.

Concerning our hiring policy, we will possibly have to challenge the attractiveness of the Paris sites. However, based on our very good relationship with the corresponding teams, we are confident that we can manage to coordinate our hiring policies in an efficient way. As proved in the past, our collaboration with these teams is always very successful.

**7.12 Perspectives**

**Keywords** Proof theory; semantics of programming languages; computer assisted proofs; Curry-Howard correspondence; linear logic; realizability; game semantics; implicit computational complexity; concurrency theory; probabilities

**Vision and new goals of the team** Based on the fruitful interaction between logic and computer science, our main objectives are directed towards extensions of the Curry-Howard correspondence to additional logical/programming paradigms. The current evolution of the team leads to a particular effort on logical approaches and to a reinforced use of abstract tools.

Our methodology is to build on mathematical theories in order to study programming languages. More precisely we want to develop new techniques from logic and semantic tools. Indeed these topics have strongly progressed during the last ten years and can now be brought together in a working framework. We plan to focus on three main mathematical tools. First, on the modelling side, game semantics has emerged as a versatile and accurate setting for representing programs and their interaction. It can handle a large spectrum of programming primitives. Second, on the reasoning side, our tool will be logic which, thanks to the proofs-as-programs paradigm, makes it possible to develop type systems and program extraction from proofs. Linear logic emerged as a powerful system to study logic(s) from this point of view. Finally, to prove the properties of our logical systems and relate them to the modelling side, we will take advantage of techniques coming from realizability, as well as from game semantics. Using these common tools, we will focus on developing type systems and program extraction mechanisms in different directions: imperative program extraction, integration of bounded complexity constraints, extension to concurrent computation, semantics of probabilistic aspects, ...

The long-term goal of establishing a unified framework (based on linear logic and related tools) for the foundations of rich programming languages (integrating concurrent, probabilistic, ... aspects with a semantic control over termination, complexity, ...) can be seen as the specificity of our team.
7.12.1 Logical foundations of programming languages

The Curry-Howard correspondence establishes a strong relation between programs with their types on the one hand, and proofs with their formulas on the other. This correspondence holds for appropriate pairs consisting of a programming language and a logic. Our three main tools (linear logic, game semantics, and realizability) are among the most advanced ones in this topic.

Linear logic has been introduced twenty years ago by J.-Y. Girard from an analysis of the semantics (denotational semantics more precisely) of intuitionistic logic and of the $\lambda$-calculus. Linear logic has had a strong influence on, and has become a key tool in various domains related to logic and computer science: from proof-theory to security protocols, through denotational semantics, and type systems for programming languages.

Originating in the idea of modelling computation using the interaction between a program and its environment, game semantics has renewed the study of denotational semantics. Its first major success has been to solve the long standing problem of giving a syntax-independent presentation of the fully abstract model (i.e. a model in which the interpretations of two programs are equal if and only if these programs have the same behaviour, as given by the observational equivalence) of PCF. Game semantics has now been successfully applied both on the logic side and on the programming side with applications to software verification.

Underlying all the computational interpretations of logical proofs, realizability can be presented as a generalization of typing. It associates programs to types in a richer way than with usual type systems, and thus can be used to “define” types from their computational contents. As a consequence, it is often at the basis of program extraction mechanisms.

Connections between these tools have already been established: game semantics was inspired from linear logic, recent models of linear logic add new perspectives in the foundations of distributed computation. The range of the programming primitives expressible in this setting is still under investigation. The study of the denotational models of differential linear logic will allow us to find abstract characterizations of behavioural equivalences of processes (a key ingredient in the theory of process algebras).

A important general question is the unification of these two key interactive interpretations of logic: game semantics and realizability. Outside constructive classical logic, we will work on understanding the computational content of “symmetric” classical systems based on the sequent calculus and dealing with a non-deterministic cut-elimination procedure.

Applications of Linear Logic

In the domain of light linear logics which are providing implicit characterizations of complexity classes, new systems have recently emerged: linear logics by levels, variations on bounded linear logic. We are interested in the syntactic study and development of these systems with two main goals. The first one is the access to additional complexity classes. The second one is to make progress on the intensional expressive power in a given class (PTIME for example): finding systems able to capture more algorithms (intensional power) for a fixed class of functions (extensional power). We also work on the denotational semantics of these variants of linear logic. By means of game semantics in particular, we hope to find models with intrinsic complexity guarantees (any proof interpretable in the model is normalizable in polynomial time for example).

The discovery of concurrent behaviours in the normalization of differential interaction nets derived from differential linear logic opens new perspectives in the foundations of distributed computation. The range of the programming primitives expressible in this setting is still under investigation. The study of the denotational models of differential linear logic will allow us to find abstract characterizations of behavioural equivalences of processes (a key ingredient in the theory of process algebras).

In the direction of unifying our approaches, a key lock will be to understand the compatibility of the variants of linear logic we use (light, differential, ...).

A long term goal is to extend Curry-Howard to computations modelled by rewriting theory in order to give them a logical content. A starting point is to use linear logic and game semantics in order to give a logical structure to residuals graphs issued form rewriting.
on these mechanisms in the future with a specific interest in extraction from classical logic (and additional axioms) and from logical systems with bounded complexity.

The development of the first classical extractor for Coq triggers important new questions: efficient execution of the obtained programs, extraction of witnesses in a large class of existential formulas, program extraction from (variants of) the axiom of choice in relation with side effects in programming. ... Using the ideas from variants of linear logic for implicit computational complexity (ICC), we will study proof systems which allow for the extraction of programs with certified complexity bounds, for instance PTIME. Our setting would add, to the usual extraction paradigm, the certification that the obtained programs admit a given complexity bound.

Realizability and program extraction are two strongly related theories. Our work on extraction in classical logic is based on Krivine’s realizability. On the ICC side, realizability techniques, as they begin to be used for light linear logics, will probably turn out to be very important.

The transfer of technologies obtained through the Curry-Howard correspondence is still under development. Standard program transformations such as terminal recursion, defunctionalization, ... have to be understood as proof transformations and the logical properties of the transformed objects should be studied. In the converse direction, focusing in logic or iterated forcing can be studied as program transformations.

7.12.2 Semantic tools for new behavioural properties of programs

Semantics (in particular denotational semantics) provides means to apply mathematical tools to the understanding of deep properties of programming languages. Results are often very strong but as a consequence they are often restricted to particular programming primitives. We want to build on recent progresses in the Curry-Howard correspondence (and related tools) to address the semantics of richer languages.

A typical application of this approach is the definition of a type system. In particular, there is a focus on so-called strong typings which, by constraining programmers to a reasonable programming discipline, provide guarantees on program behaviours.

We will focus mainly on implicit complexity constraints on the execution of programs, concurrent behaviours of computational systems, and randomized aspects of languages. This naturally leads us to address new research directions such as the control of complexity properties of concurrent processes.

ICC

We want to work on the expressiveness of implicit complexity criteria. For this purpose, we will study, given a criterion, the behavioural properties of the programs satisfying the criterion. For instance we can try and investigate necessary (and, possibly, sufficient) conditions on the behaviour of abstractions of these programs. The purpose of this approach is on the one hand to reveal theoretical limitations of certain criteria, by showing that certain families of programs cannot be validated, and, on the other hand, to compare different criteria.

In order to design realistic functional languages with bounds on usage of resources, a crucial point is to combine a usable but constrained form of recursion with a rich pattern-matching mechanism. To address this question, we envisage to mix typing constraints coming from light linear logics with other termination criteria for higher-order recursion such as sized-based termination.

Concurrency

Regarding the theory of process calculi, which provides formal models of concurrent and mobile computation, we will continue working on the behavioural properties of processes. Questions we plan to address include algebraic characterizations of observational equivalences, the definition of methods for the guarantee of behavioural properties, and of proof techniques for reasoning about concurrent systems.

A new planned research direction stems from the interaction with ICC for sequential computation. We aim at investigating techniques to guarantee quantitative bounds on concurrent systems, by designing suitable type systems or static criteria. For instance we would like to introduce criteria to guarantee that after an interaction, a system will react after a certain number of internal steps, or that during execution, the size of the system will remain bounded. Such properties are clearly relevant in a concurrent setting: for instance, one would like a request to a server to be answered after a reasonable number of computation steps, or that an agent does not use too many resources for too much time.

Randomized aspects of computation

Adding probabilistic traits to imperative languages has been broadly studied. Among the main references on this topic, the more related to our source of interest include works done by Kozen, Morgan and Mc Giver. When it comes to functional paradigm though, one has to deal with new objects: so called high order types, hence functional spaces.

On a practical side, the consequence is one could provide programs which generate randomized functional terms; and this is different from providing programs which return a randomized value form any given value for the input parameter(s)! The theoretical consequence is that we must understand which mathematical meaning (semantics) can be given to functional spaces in the first place.

This correspondence between programs and mathematical objects matches the broader Curry-Howard correspondence, for which experience shows that the actual understanding of probabilistic traits should benefit from exploring side by side both practical and theoretical issues. Therefore, our research program in the area is twofold: developing semantics (the adaptation of mathematical objects taken from functional analysis like Polish spaces and similar ones looks promising); and proceed forward building formal tools to specify and
prove properties on actual programs, as we are confident this approach could benefit from programmer’s needs and/or understanding on randomized algorithms. Also, research areas such as Cryptography and Complexity could contribute to provide deeper insight on that matter.

7.12.3 Formalization

Proof assistants and formal logics are not only an object of study for us. We also use them for the formalization of mathematical theories. Formalization is often important and useful to clarify the axioms and reasonings being used, and to find appropriate ways to define the objects being manipulated. Currently our developments are made with the Coq system.

Our formal study of game theory and Nash equilibria will be continued with applications to the Internet (infinite games) and to biology (feasibility-desirability games and coalition games).

The formalization of Euclidean plane geometry will be turned into a tool to help pupils in the solving of geometric problems by providing dynamic drawings and proof verifications. This requires good interaction between the drawing tool and the proof assistant, through the definition of specific tacticals.

We also use Coq for proving properties of various algorithms. Examples that we plan to address include watermarking algorithms, distributed and/or randomized algorithms.

7.13 Publications

International and national peer reviewed journals [ACL]


Invited conferences, seminars, and tutorials [INV]

International and national peer-reviewed conference proceedings [ACT]


**Short communications [COM] and posters [AFF] in conferences and workshops**


[646] Romain Demangeon. Type systems for the termination of mobile processes. talk given at the 10th International Workshop on Termination, Leipzig, Germany, 2009.


[649] Barbara Petit. A polymorphic type system for the lambda-calculus with constructors. talk at the TYPES’09 meeting, Aussois, France, 2009.

**Scientific books and book chapters [OS]**


**Book or Proceedings editing [DO]**


Other Publications [AP]

Doctoral Dissertations and Habilitation Theses [TH]


<table>
<thead>
<tr>
<th></th>
<th>2006</th>
<th>2007</th>
<th>2008</th>
<th>2009</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL - International and national peer-reviewed journal</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>INV - Invited conferences</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ACT - International and national peer-reviewed conference proceedings</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>19</td>
</tr>
<tr>
<td>COM / AFF - Short communications and posters in international and national conferences and workshops</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>OS - Scientific books and book chapters</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>OV - Scientific popularization</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>DO - Book of Proceedings editing</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>AP - Other Publications</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TH - Doctoral Dissertations and Habilitation Theses</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>10</td>
<td>11</td>
<td>16</td>
<td>18</td>
<td>55</td>
</tr>
</tbody>
</table>

Other Productions [AP]


