INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team LogiCal

Logic and Computation

Futurs

THEME SYM

Table of contents

1. Team

The LogiCal project is a common project gathering researchers from INRIA-Futurs at LIX and Laboratoire de Recherche en Informatique of University Paris XI.

Scientific leader
Gilles Dowek [Professor, Ecole Polytechnique, HdR]

Vice leader
Benjamin Werner [CR INRIA]

Administrative assistant
Catherine Moreau [TR INRIA]

INRIA staff
Bruno Barras [CR INRIA]
Hugo Herbelin [CR INRIA, HdR]

Paris XI staff
Jean-Pierre Jouannaud [Professor, Université Paris-Sud, HdR]

CNRS staff
Jean-Marc Notin [IR CNRS]

Post-doctorates
Evgeny Makarov
Gyesik Lee

Ph.D. students
Lisa Allali [Région Ile-de-France (since march)]
Denis Cousineau [MENRT (since march)]
Dan Hernest [France-Télécom(until december 15)]
Florent Kirchner [École polytechnique]
Sylvain Lebresne [MENRT]
Julien Narboux [MENRT (until september)]
François-Régis Sinot [École polytechnique (until september)]
Elie Soubiran [MENRT (since march)]
Arnaud Spiwack [ENS Cachan (since march)]
Pierre-Yves Strub [EADS]
Roland Zumkeller [MENRT]

2. Overall Objectives

2.1. Overall Objectives

Many human activities have been transformed by the invention of the computer and its broad diffusion in the second half of the XX$^{\text{th}}$ century. In particular, mathematicians could, from then on, use a tool allowing to carry out operations that were too long or to tedious to be executed by hand. Like the use of the telescope in astronomy, the use of the computer opened many new prospects in mathematics. One of these prospects is the use of *proof assistants*, *i.e.* computer programs which perform some operations on mathematical proofs. The goal of the research developed in the LogiCal project-team is to develop such *proof assistants*. The main effort the project-team is the development of the Coq system, which has an important community of users in industry and in academia. However, we believe that the development of a proof assistant cannot be accomplished without a joint reflection about the structure of mathematical proofs and about the use of proof assistants in various applicative domains. Thus, the questions addressed in the team range from questions related to the Coq system, such as “What will be the features of the next version of Coq?”, to more theoretical questions of logic, such as “What is a proof?” and more applied ones, such as “How can we use a proof assistant to check a protocol if free of deadlocks?”.

3. Scientific Foundations

3.1. Proof assistants

Keywords: *correctness*, *proof assistant*, *tactic language*.

The first operation that a proof assistant can perform on a proof is to check its correctness. This participates in the quest of a new step in mathematical rigor: the point where nothing is understated, and where the reader can therefore be replaced by a program. This quest for rigor is specially important for the large proofs, either hand written or computer aided, that mathematicians have built since the middle of the XX^{th} century. For instance, without using a proof assistant, it is quite difficult to establish the correctness of a proofs using symbolic computations on polynomials formed with hundreds of monomials, or a case analysis requiring the inspection of several hundreds of cases, or establishing that a complex object such as a long program or a complex digital circuit has some property. This quest for correctness is especially important in application domains where a malfunction may jeopardize human life, health or environment, such as transportations or computer aided surgery.

Besides this correctness check, proof assistants can help the users to build proofs interactively. The "tactic language" allowing the user to control the system in this proof construction process has always been the object of intensive studies. The ML language, for instance, was originally the tactic language of the LCF proof assistant. More recent questions about this language are focused on the formal expression of its operational semantic, in particular the handling of exceptions.

Proof assistants may also prove some easy lemmas automatically, transform mathematical proofs into other formal objects such as programs.

A more recent kind of applications is the construction of large libraries of mathematical results on the net.

3.2. Formalization of mathematics

Keywords: *Calculus of Constructions*, *constructive proofs*, *deduction modulo*, *mathematical language*, *predicate logic*, *programming language*, *set theory*, *type theory*.

A proof assistant implements a particular formalism allowing to express mathematics. A traditional formalism allowing to express mathematics is set theory, built on top of first-order predicate logic. Unfortunately, this formalism does not address exactly the needs of a proof assistant. Set theory has been elaborated at the beginning of the XX^{th} century to study mathematically the properties of mathematical reasoning. For this purpose, being able to formalize mathematics "in principle" was enough. Nowadays, the problem is not to formalize mathematics "in principle" but to formalize them "in facts". Thus, the design of proof assistants has led to ask new questions in logic and, in particular, in proof theory.

Several variants or alternative to set theory have been designed to express mathematics in practice. The system Coq is based on a formalism called *The Calculus of Inductive Constructions*.

An important feature for such a formalism is the language allowing to express mathematical objects such as functions and sets. It is not possible to use a formalization of mathematics that has only existence axioms, or even one having the combinator's language obtained by skolemizing these axioms in predicate logic. It is important to have a rich and compact language, in particular a language with binders such as the λ-calculus.

Another important feature is the ability to integrate deduction and computation. It is not possible, when we use a proof assistant to consider that the proposition $2 + 2 = 4$ requires a proof, even a proof simple enough to be found by a automated theorem proving system. Several formalisms such as Martin-Löf's type theory, Boyer-Moore logic, the Calculus of Constructions and the Calculus of Inductive Constructions, include such a possibility to compute inside a proof. Thus, these formalisms designed to express mathematics contain a programming language as a sub-language.

More recently the research in this area has taken several different directions: first the study of *deduction modulo* that is the simplest extension of predicate logic allowing to mix deduction and computation. Deduction modulo has applications both in automated theorem proving and in proof theory, where it paves the way to a unified theory of cut elimination. Another direction is the design of extensions of the Calculus of Constructions with arbitrary computation rules, while the original calculus had a fixed set of rules. This extension called the *Calculus of Algebraic Constructions* may be the future formalism used in the Coq system. Finally, the need to improve the efficiency of computations in the system Coq, has led to the use of compilation techniques issued from the theory of programming language. This has brought logical languages and programming languages closer, allowing for instance to use the language of Coq as a general purpose programming language. This perspective of unifying languages and programming languages is a real challenge for future proof assistants.

Another property of the Calculus of Inductive Constructions is important for its use as the language of a proof assistant. The first is the possibility to write both constructive and classical proofs. When a proof of existence is constructive, the user can request the computation of a witness, but, of course, not when it is classical.

By insisting on this idea that constructive proofs must be distinguished from classical proofs, the project-team LogiCal participates to rise of a new form a constructivism, not trying to restrict mathematics to constructive mathematics, but trying to identify the part of mathematics that can be done constructively and the part that cannot.

A last property of the Calculus of Inductive Constructions is that proofs are objects of the formalism, exactly as numbers, functions and sets are. This property, based on the celebrated Curry-De Bruijn-Howard correspondence, allows to reduce the safety critical base of the Coq system to a quite small kernel.

4. Application Domains

4.1. Application Domains

Keywords: *algorithms*, *mathematics*, *programs*.

The applications of the research of the LogiCal project-team take several directions.

The first is the applications to pure mathematics. The use of proof assistants for proving genuine mathematical theorems has been considered as utopic for long. But several recent developments have changed the situation. First of all, the development of libraries of both constructive and classical analysis has led the possibility to use Coq, not only in remote areas of discrete mathematics, but also to prove mainstream mathematical theorem as taught in an undergrad textbook for instance. This direction culminated with the proof in Coq of the Fundamental Theorem of Algebra, a few years ago, by a group of researchers in Nijmegen. More recent work include a proof of the Four color theorem in Coq. Proofs of lemma's on polynomials used in the proof of Hale's Sphere packing theorem (Kepler's conjecture) and proofs in algebraic geometry by a group of mathematicians in Nice.

Another direction is the proof of algorithms. In proofs of algorithms (as opposed to proofs of programs) a property is proved on an algorithms formalized in the language of Coq. An example is the recent proof of algorithms used in floating point arithmetic or the older proof carried out by the company *Trusted Logic* of the correctness that has reached, for the first time, the EAL7 level in common criteria.

But, our main application domain is the proof of programs where an actual program written in the syntax of a general purpose programming language (such as Caml, Java or C). The system Coq is used by the ProVal project-team, that has strong historical connections to LogiCal, as a back-end of their systems Why, Krakatoa and Caduceus.

5. Software

5.1. Coq

Participants: Bruno Barras, Jean-Christophe Filliâtre, Hugo Herbelin, Christine Paulin.

The *Coq* system, developed in the project, is a processor of mathematical proofs allowing an interactive development of specifications and proofs. The main original aspect of the *Coq* system is its formalism that includes:

- a primitive notion of mutual inductive definitions allowing high level specification either in a functional style by declaring concrete datatypes and defining functions by equations representing computations, or in a declarative style by specifying relations thanks to clauses;
- an interpretation of proofs as certified programs, implemented by the compilation of proofs as ML programs but also tools to associate a program to a specification and automatically generate proof obligations to assert its correctness;
- a primitive notion of co-inductive definitions allowing a direct representation of infinite rational data structures and build proofs upon such objects without resorting to the classical notion of bisimulation.

At the architectural level, the main features are:

- an interactive loop that allows to define mathematical and computational objects and to state lemmas,
- the interactive development of proofs thanks to a large and extendable set of tactics that decompose into elementary tactics (giving a precise control over the proof structure and thus over the underlying program) and decision or semi-decision procedures.
- a modular standard library and retrieving tools,
- a mechanism to perform partial or total evaluation of programs written within the language of *Coq*,
- a module system to manage name spaces, and featuring functors to develop parameterized development and making easier the instantiation of such functors,
- the possibility to develop evolved tactics written in the implementation language of *Coq* (namely Objective Caml), and that can be dynamically loaded and used from the toplevel,
- the isolation of the critical code preforming the proof checking in a kernel small enough to reach higher levels of reliability of the whole system (with the current goal of achieving the self-validation), and the production of an abstract interface of that kernel granting that theories can only be built using the features of the kernel.

Among the most significant achievements realized using *Coq*, it worths mentioning:

- the model of authentication protocol CSET used in electronic shopping and the proof of properties of this protocol,
- the correctness proof of a compiler of the reactive language Lustre, used in the industrial setting of Scade,
- a proof of the critical kernel of the *Coq* environment,
- several models of the properties of the π-calculus,
- the development of libraries about algebra, analysis and geometry,
- a certified version of Buchberger's algorithm used in computer algebra,
- the proof of FTA theorem,
- the proof of Taylor's approximation theorem,
- the proof of the Four color theorem.

5.1.1. New features of Coq

Hugo Herbelin supervised the development of the Coq system. Beta-versions of Coq version 8.1 have been released in June and November 2006. Hugo Herbelin implemented various new tactics and features for Coq. The most significative one is an implementation of "universe polymorphism for inductive types" that allows for more sharing of data structures in Coq.

5.1.2. The Coq product

The *Coq* system is available from URL http://coq.inria.fr/. Written in Objective Caml and Camlp4, it is ported to most Unix architectures, but also to Windows and MacOS.

Coq is used in hundreds of sites. We have demanding users in industry (France Télécom R & D, Dassault-Aviation, Trusted Logic, Gemplus, Schlumberger-Sema, ...) in the academic world in Europe (Scotland, Netherlands, Spain, Italy, Portugal, ...) and in France (Bordeaux, Lyon, Marseille, Nancy, Nantes, Nice, Paris, Strasbourg, ...).

An electronic mailing list (mailto:coq-club@pauillac.inria.fr) fosters exchange between persons interested by the system.

5.1.3. GeoProof

Participant: Julien Narboux.

GeoProof is available on the web (http://home.gna.org/geoproof/)

GeoProof is an interactive geometry software with proof related features. The project consist in producing an interactive proof software for geometry. GeoProof can communicate with the Coq proof assistant to perform automatic and interactive proofs of geometry theorems.
The main features are:

- computations are done using arbitrary precision
- some theorems can be checked using the Wu and Gröbner automated theorem proving methods
- GeoProof can communicate with CoqIde (a user interface for Coq). The user can build a construction using GeoProof and the corresponding formula is automatically translated into Coq's syntax.

6. New Results

6.1. Development of theories and tactics

6.1.1. Hales' Theorem

Participants: Roland Zumkeller, Benjamin Werner.

Roland Zumkeller has worked on a contribution to the formalization of Thomas Hales' proof (1998) of the Kepler conjecture.

In order to prove a list of some thousand inequalities appearing in this proof he has implemented an algorithm based on Taylor models in Coq, described in [27]. In principle this makes any strict real inequality amenable to formal verification, even when elementary functions are involved.

In order to make this practical, a better method for polynomial optimization is still needed. To this end Roland Zumkeller has studied the method of rewriting a polynomial as a sum of squares, based on relatively recent work by Pablo Parrilo. In December he visited the National Institute for Aerospace in Langley, Virginia, where he implemented a simple tool for sum of squares representation [which has been integrated into PVS by César Muñoz].

Roland Zumkeller attended TYPES 2006 in Nottingham, England, where he gave a talk.

Roland Zumkeller attended IJCAR 2006 in Seattle, Washington, where he gave a talk.

6.1.2. Prime Numbers in Type Theory

Participants: Benjamin Werner, Benjamin Grégoire, Laurent Théry.

Benjamin Werner, Benjamin Grégoire and Laurent Théry have presented a new way to treat Pocklington primality certificates in Type Theory. This was implemented in Coq [16] and allows to prove the primality of numbers of over 1000 digits.

6.1.3. Proofs in geometry

Participants: Julien Narboux, Hugo Herbelin.

Julien Narboux performed the mechanisation of the proofs of the first height chapters of Schwabäuser, Szmielew and Tarski's book: *Metamathematische Methoden in der Geometrie*. The goal of this development is to provide foundations for other formalizations of geometry and implementations of decision procedures. He compared the mechanized proofs with the informal proofs. He also compared this piece of formalisation with the previous work done about Hilbert's *Grundlagen der Geometrie*. The differences between the two axiom systems from the formalisation point of view are analysed. A paper has been accepted at Automated Deduction in Geometry 06 [24].

6.2. Development of systems

6.2.1. Coq 8.1

Participants: Hugo Herbelin, Bruno Barras.

Hugo Herbelin provided support for the version 8 of Coq. See the coq web site for further details.

Bruno Barras coordinated the release of release of Coq 8.1.

6.2.2. New field tactic

Bruno Barras integrated a new implementation of the field tactic contributed by Laurent Thery. This tactic solves or simplifies field equations, provided the user proved that the domain of the equation satisfies the field axioms. The main task was to provide a better interoperability with the previously reimplemented ring tactic, which provides the same features but on ring structures. The usage of the Ltac language allowed to reduce significantly the need for ML code dedicated to the tactic. Practically, it also allows users to implement other variants of the tactic without writing ML code.

6.2.3. Syntactic guard check of termination of recursive definitions

Bruno Barras has studied the syntactic guard criterion, which is the part of kernel of Coq checking that recursive functions (fixpoints) always terminate. He wrote a definition of that part of the kernel, which extends significantly all previously studied systems. A critical implementation error was found and fixed. Finally, he made a survey on the various ways of proving the correctness of this syntactic criterion, and possible alternative implementations of a termination checker based on typing.

6.2.4. GeoProof

Participants: Julien Narboux, Hugo Herbelin.

Julien Narboux has designed of a graphical user interface to deal with proofs in geometry. The software developed (GeoProof) combines three tools: a dynamic geometry software to explore, measure and invent conjectures, an automatic theorem prover to check facts and the interactive proof system Coq to mechanically check proofs built interactively by the user. A paper have been accepted at the Journal of Automated Reasoning special issue on User Interfaces for Theorem Provers [10].

6.2.5. Fellowship

Participants: Florent Kirchner, Gilles Dowek.

Florent Kirchner has been working on the implementation of the proof management software Fellowship. Prominently, he developped a library of Real analysis inspired by Coq's, and generated from it code for both Coq and PVS. This has entailed the study of a finite first-order theory of classes, which has recently been accepted for publication in TYPES [21].

6.2.6. The Calculus of Congruent Constructions

Participants: Pierre-Yves Strub, Jean-Pierre Jouannaud.

Pierre-Yves Strub has developed, in the OCaml language, a proof checker for the *Calculus of Congruent Constructions*; and next, a proof editor (which includes a tiny subset of the tactics available in the Coq system) allowing users to develop proofs in the new calculus. In order to allow the use of the Maude2 system in other programming languages, Pierre-Yves Strub has implemented a module which drives the Maude2 system by the way of a XML protocol. An OCaml module which implements this protocol has been developed too.

6.2.7. Decision procedures

Participants: Pierre-Yves Strub, Jean-Pierre Jouannaud.

Pierre-Yves Strub has implemented in the Maude2 system (Maude is a reflective language, influenced by the OBJ3 language, supporting both equational and rewriting logic specification and programming for a wide range of applications. See http://maude.cs.uiuc.edu/) a module which implements the Shostak algorithm for combining decision procedures, for the equality in first order logics, into a general one. He also has implemented, in the same system, decision procedures for linear arithmetic and algebraic data types.

6.2.7.1. Generic proof languages

Participants: Florent Kirchner, Gilles Dowek, César Muñoz.

Florent Kirchner has been working on the implementation of the proof management software Fellowship. Prominently, he developped a library of Real analysis inspired by Coq's, and generated from it code for both Coq and PVS. This has entailed the study of a finite first-order theory of classes, which has recently been accepted for publication in TYPES [21].

Florent Kirchner has maintained a strong interest in the semantics of imperative programming languages in general, and in proof languages in particular. He co-authored with François-Régis Sinot a paper accepted at RULE'06 [23], demonstrating on a simple example how novel techniques could be adapted to the formulation of such semantics. He also extended an earlier work on the topic [20] and submitted it for journal publication.

Florent Kirchner has collaborated with César Muñoz on the topic of PVS's proof language, resulting in an accepted submission at STRATEGIES'06 [22]. Work has also begun on the formalization of the mathematical basis of proof languages, as well as on the more practical, methodological approaches for their designs.

6.3. Studies of formalisms

6.3.1. Diagrammatic proofs in abstract rewriting

Participant: Julien Narboux.

Julien Narboux has studied the kind of diagrams which can be found in the rewriting community. He gave a formal definition of the diagrams which are used to state properties and proposed inference rules to formalise some diagrammatic proofs such as the proof of the Newman's lemma. He showed that the system proposed is both correct and complete for a class of formulas called "coherent logic". This work has been submitted as a journal paper [35].

6.3.2. Proof-irrelevant theories

Benjamin Werner has presented a version of type theory where non-computational proof-terms are not relevant for conversion check anymore [26].

Bruno Bernardo and Bruno Barras have started a more ambitious effort in a similar direction by proposing a version of Alexandre Miquel's Calculus of Implicit Constructions with decidable type-checking.

6.3.3. Deduction modulo

Participants: Olivier Hermant, Gilles Dowek, Benjamin Wack.

Gilles Dowek has proposed a new semantic for deduction modulo where propositions are interpreted in truth values algebras, that generaliz Heyting algebras. He has defined a notion of super-consistent theory and proved, using a previous work, common with Benjamin Werner that super-consistent theories had the normalization property.

Gilles Dowek and Olivier Hermant, have then given two simpler proofs that a super-consistent theory had the cut elimination property. Although these proofs are quite similar, they seem to have different computational content. The content of the first being a proof-normalization algorithm and that of the second a proof-search algorithm.

Paul Brauner, Gilles Dowek and Benjamin Wack have started a work dedicated to the comparison of deduction modulo with super-natural deduction.

Lisa Allali proposed a new formulation of arithmetic as a theory modulo without axioms. This work is based on the study of equality in Heyting Arithmetic and more specificaly how to use the decidability of equality in Arithmetic to find an algorithm using rewriting rules to decide equality. This theory doesn't need anymore the Leibniz Axiom ($\forall x \forall y\ x = y \Rightarrow P(x) \Rightarrow P(y)$) to define equality. Equality is defined by a set of rewriting rules. This new definition improves an existinig modulo theory of Heyting Arithmetic purely comptable (without any axiom). This theory is a conservative extension of Heyting Arithmetic. Moreover it has cut elimination property.

6.3.4. Lambda-calculus modulo

Participants: Pierre-Yves Strub, Jean-Pierre Jouannaud, Denis Cousineau, Gilles Dowek.

Pierre-Yves Strub, Jean-Pierre Jouannaud and Frédéric Blanqui have worked on the Calculus of Congruent Constructions which replaces the conversion rule of the traditional Calculus of Constructions by a much stronger version checking whether the equality of two formulae is implied by some information collected from the context of the proof. This mechanism is indeed inspired from Shostak's combination of decision procedures, which has been proved very useful in PVS. This work is now ready for submission. Apart from a new implementation of Coq, a further step of the work includes replacing Shostak's method by a more recent one called DPLL in which the implication used by Shostak is replaced by an arbitrary propositional formula.

Denis Cousineau started a Ph.D thesis this year, on normalisation in Lambda Pi calculus modulo. The dependently typed lambda-calculus, or lambda-Pi-calculus, allows to express proofs of minimal first-order predicate logic. It can be extended in a very simple way, by adding computation rules. This leads to the lambda-Pi-calculus modulo. Denis Cousineau and Gilles Dowek proved in [34] that this simple extension is surprisingly expressive and, in particular, that all functional Pure Type Systems, such as the system F, or the Calculus of Constructions can be embedded in it in a conservative way.

6.3.5. Director strings and proof nets

Participants: François-Régis Sinot, Jean-Pierre Jouannaud, Ian Mackie, Maribel Fernandez.

François-Régis Sinot pursued his work on the relationships between traditional abstract machines based on environments and interaction nets and published an extended journal version [11] of his earlier work [12].

François-Régis Sinot is also a co-author of a paper linking the rewriting calculus, interaction nets and strategies in functional programming languages [6], as a result of his involvement in an Alliance project.

However, most of François-Régis Sinot's efforts this year were devoted to writing up and defending his PhD thesis [4]. In addition to the directors, Maribel Fernández and Jean-Pierre Jouannaud, the jury was composed of two reviewers, René David and Vincent van Oostrom, and two examiners, Martin Hyland and Jean-Jacques Lévy.

6.3.6. Inductive types

Participants: Sylvain Lebresne, Hugo Herbelin.

Sylvain Lebresne has proposed a coding of the dependent elimination schemes of inductive data types using Σ-types.

6.3.7. Termination

Participants: Jean-Pierre Jouannaud, Albert Rubio, Frédéric Blanqui.

Jean-Pierre Jouannaud, Albert Rubio and Frédéric Blanqui have continued their work on the higher-order recursive path ordering for proving termination of higher-order rules that use plain pattern matching in a setting with weak higher-order polymorphic rules. This ordering includes beta- and eta-reductions, is well-founded, and polymorphic in the sense that a single comparison allows to prove termination of all monomorphic instances of the rule. The paper to be published in JACM is far more advanced that their previous version published at LICS'99 and allows to deal with most examples found in the literature. A PROLOG implementation was written and is promising.

6.3.8. Confluence

Participants: Jean-Pierre Jouannaud, Albert Rubio, Femke Van Raamsdonk, Yoshihito Toyama.

Jean-Pierre Jouannaud, Albert Rubio and Femke Van Raamsdonk have worked on the problem of proving confluence of terminating higher-order rules in the same type setting again. They have described a general abstract framework called *normal rewriting* which can then be instantiated in various ways, depending on the pattern matching algorithm in use, in order to compute the appropriate critical pairs for each case. This work is currently being rewritten after an unsuccessful submission.

Jean-Pierre Jouannaud has also worked on Toyama's theorem for modular confluence, whose proof has remained complex until now. He found a new proof based on a modularity property of ordered completion which is easy to grasp and teach. He also has generalized the result to rewriting modulo an arbitrary equational theory, for all known (and yet unknown!) variations of rewriting modulo thanks to a simple remark made by Toyama. This work is submitted to the next RTA conference.

6.4. New Computation Paradigms

6.4.1. Classical Logic

Participants: Hugo Herbelin, Sylvain Lebresne, Zena Ariola, Amr Sabry, Mircea-Dan Hernest.

Sylvain Lebresne worked on classical logic, exploring the possibility of mixing call-cc style control operator with dependent types, especially with sigma types. He then started studying exceptions in programming languages, proposing a way of adding exceptions to the calculus of constructions. One of the challenges is to investigate a call-by-name vision of exceptions. This work is still in progress.

Mircea-Dan Hernest has continued his work on the complexity of programs extracted by means of Gödel's functional interpretation and its monotone variant due to Kohlenbach. His research materialized by his PhD, defended in, december. He gives a new adaptation of Gödel's technique to the extraction of more efficient programs from Natural Deduction arithmetical proofs. He also finalized his joint paper with Kohlenbach on the computational complexity of the monotone Dialectica extraction algorithm and started to design a framework for the extraction of poly-time computable bounds by the monotone Dialectica interpretation.

6.4.2. Quantum computation

Pablo Arrighi and Gilles Dowek have proposed a higher-order extension of their linear-algebraic lambda-calculus and proved the confluence of its semantic. They have shown that this notion of higher-order permits to express in a very simple way black-box algorithms, such as Deutsch-Josza algorithms.

6.4.3. Plan execution language

C. Muñoz, G. Dowek, and C. Pasareanu have defined a semantic framework that allows to define plan execution langages, such as the language PLEXIL used for planning spacecraft missions.

7. Contracts and Grants with Industry

7.1. Mao

MAO is an ACI (ministry grant) about developing an interface and libraries on top of Coq in order to provide support for "professional mathematicians". It gathers both computer scientists (projects LogiCal and Marelle) and mathematicians (Lab. Dieudonné, University of Nice). The project's homepage URL is http://math1.unice.fr/~jpg/aci/index.htm

7.2. France Telecom

The project has a a three year contract with France Télécom.

7.3. EADS

The project has a a three year contract with EADS.

8. Other Grants and Activities

8.1. Collaboration with other teams

LogiCal has a strong link with the new INRIA-Microsoft Research joint Laboratory, of which Roland Zumkeller, Benjamin Werner and Bruno Barras are also members.

François-Régis Sinot and Ian Mackie collaborate actively with the Theory of Computing group at King's College (London). LogiCal has also active collaborations with other INRIA projects: Marelle, Cristal, Protheo, ProVal.

8.2. European actions

8.2.1. Working Group TYPES

Working Group "TYPES" is about computer aided development of proofs and programs.

It is composed of teams from Helsinki, Chambéry, Paris, Lyon, Rocquencourt, Sophia Antipolis, Orsay, Darmstadt, Freiburg, München, Birmingham, Cambridge, Durham, Edinburgh, Manchester, London, Sheffield, Padova, Torino, Udine, Nijmegen, Utrecht, Bialystok, Warsaw, Minho, Chalmers, and also from Prover Technology, France Télécom, Nokia, Dassault-Aviation, Trusted Logic and Xerox companies.

For LogiCal, Benjamin Werner acts as a site leader for a group of subsites including Sophia-Antipolis, Bologna, Dassault-Aviation and Minho.

8.2.2. Alliance project

François-Régis Sinot is involved in an Alliance project on Implementation Techniques for the Rewriting Calculus, including several people from the LogiCal INRIA Futurs project, the Protheo INRIA Loria project and the Theory of Computing group at King's College London.

9. Dissemination

9.1. Animation of the scientific community

9.1.1. Editorial charges

Benjamin Werner co-organized a TYPES affiliated workshop on proofs and numbers, in the frame of the INRIA - Microsoft joint lab.

9.1.2. Committees

Benjamin Werner has been member of the program committee of the *Journées Francophones des Langages Applicatifs* 2006.

Hugo Herbelin has been a member of the program committee of the workshops "Classical Logic and Computing" (CL&C, July 2006, Venice, Italy), "Strategies in Automated Deduction" (STRATEGIES, August 2006, Seattle, USA) and "Programming Languages meets Program Verification" (PLPV, August 2006, Seattle, USA).

Gilles Dowek has been a reviewer of the Doctoral Dissernation of Stéphane Lengrand

Gilles Dowek has been a reviewer of the Doctoral Disseration of Frédéric Ruyer.

Hugo Herbelin was external referee for Samuel Howse's PhD examination committee (Halifax, Canada, October 2006).

Benjamin Wernerserved on Assia Mahboubi's PhD comitee.

9.1.3. Visits

Gilles Dowek has visited twice César Muñoz at the National Institute of Aeropsace (Hampton, Virginia, US) Florent Kirchner visited the National Institute for Aerospace during the months of July and September. Roland Zumkeller visited the National Institute for Aerospace in december.

Gilles Dowek has given a serie of popular science conference in Quebec including a "Bar des Sciences" in Jonquière, talks for the students of several high-school, Cegep and Universities in Montreal, Jonquière and Chicoutimi.

Gilles Dowek has visited the University of Ottawa where he has given a lecture.

Julien Narboux has given a talk at the seminar "Géométrie Algébrique" in the laboratory Dieudonné in Nice, June 2006.

Julien Narboux has given a talk at the seminar of the team Marelle at INRIA Sophia Antipolis, June 2006.

Julien Narboux has given two talks at the Club2 seminar of TU Munich, October 2006.

9.1.4. Conferences

Gilles Dowek has given a talk at the meeting on *Modern Type theory* at the Institut d'Histoire et de Philosophie des Sciences et des Techniques, March 24th and 25th, 2006.

Gilles Dowek has given a course at the "International Summer School on Rewriting" in Nancy.

Gilles Dowek has participated to the meeting "Logique et Interaction - Géométrie de la Cognition" in Cerisy where he has given a talk.

François-Régis Sinot has attended the Third Workshop on the Rho-Calculus in London.

François-Régis Sinot has attended the TYPES'06 workshop in Nottingham.

François-Régis Sinot has given talks at the Term Rewriting Seminar (TeReSe) in Amsterdam, at the Logic Group Seminar of LAMA in Chambéry, at the University of Porto and at the PPS seminar at University Paris 7.

Hugo Herbelin attended the CL&C workshop in Venice, Italy (July 2006) where he gave an invited talk.

Julien Narboux has attended the 6th Automated Deduction in Geometry International Workshop where he has given a talk [24].

Hugo Herbelin, Florent Kirchner, Dan Hernest Mircea, Julien Narboux, Roland Zumkeller and Benjamin Werner have attended to various conferences or workshops of the FLOC 2006 event in Seattle. Various talks were given: (Spiwack: LICS, Werner and Zumkeller: one talk at IJCAR each, Herbelin: HOR, Narboux: UITP, Kirchner: RULE and STRATEGIES, Mircea: RULE).

Gilles Dowek, Hugo Herbelin, Benjamin Werner, Roland Zumkeller, FranÃ§ois-Régis Sinot, Florent Kirchner and Sylvain Lebresne have attended the TYPES '06 conference in Nottingham. Talks were given by Dowek, Werner and Zumkeller.

Hugo Herbelin and Bruno Barras have attended to the CHIT and CHAT TYPES-affiliated workshops in Nijmegen.

Pierre-Yves Strub, Sylvain Lebresne and Julien Narboux have attended to the Marktoberdorf'06 Summer School on *Logical Aspects of Secure Computer Systems*.

9.1.5. Other charges

Jean-Pierre Jouannaud is the leader of the LIX laboratory. He is president of AFIT, and member of "council of ETACS".

Bruno Barras is consultant in formal methods at Trusted Labs, located in Versailles.

Benjamin Werner has been invited to record a talk on computational proofs, now available on INRIA's main web site.

Florent Kirchner and Julien Narboux are the web-masters of the Coq and LogiCal web sites.

9.2. Teaching

Gilles Dowek is the thesis advisor of Florent Kirchner, Denis Cousineau and Lisa Allali. Hugo Herbelin is the thesis advisor of Elie Soubiran and co-advisor of Sylvain Lebresne. Benjamin Werner is thesis advisor of Roland Zumkeller and co-adviser of Arnaud Spiwack. Bruno Barras is thesis advisor of Bruno Bernardo. Jean-Pierre Joouannaud is thesis advisor of Pierre-Yves Strub.

Gilles Dowek has given a course at the Markoberdorf Summer School and at the TYPES Summer School.

Benjamin Werner and Hugo Herbelin have given courses in the *Master Parisien de Recherche en Informatique*.

Julien Narboux has been teaching assistant at the University Paris XI.

10. Bibliography

Year Publications

Books and Monographs

[1] K. FUTATSUGI, J.-P. JOUANNAUD, J. MESEGUER (editors). *Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, Lecture Notes in Computer Science, vol. 4060, Springer, 2006.

[2] F. PFENNING (editor). *Term Rewriting and Applications, 17th International Conference, RTA 2006, Seattle, WA, USA, August 12-14, 2006, Proceedings*, Lecture Notes in Computer Science, vol. 4098, Springer, 2006.

Doctoral dissertations and Habilitation theses

[3] J. NARBOUX. *Formalisation et automatisation du raisonnement géométrique en Coq*, Thése de doctorat, spécialité informatique, Université Paris-Sud, September 2006.

[4] F.-R. SINOT. *Efficient Strategies and Implementation Models for Functional Languages*, Thése de doctorat, spécialité informatique, Ecole Polytechnique, école Polytechnique, France, September 2006.

Articles in refereed journals and book chapters

[5] F. BLANQUI, J.-P. JOUANNAUD, A. RUBIO. *Higher-Order Termination: from Kruskal to Computability*, Invited lecture, vol. 4246, 2006.

[6] H. CIRSTEA, G. FAURE, M. FERNÁNDEZ, I. MACKIE, F.-R. SINOT. *New Evaluation Strategies for Functional Languages*, in "Electronic Notes in Theoretical Computer Science", to appear, 2006.

[7] G. DOWEK, Y. JIANG. *Eigenvariables, bracketing and the decidability of positive minimal predicate logic*, in "TCS", vol. 360, 2006, p. 193-208.

[8] J.-P. JOUANNAUD, A. RUBIO. *Polymorphic Higher-Order Recursive Path Orderings*, in "JACM", to appear 2007, 2006.

[9] J.-P. JOUANNAUD, W. XU. *Automatic Complexity Analysis for Programs Extracted from Coq Proof.*, in "Electr. Notes Theor. Comput. Sci.", vol. 153, n^{o} 1, 2006, p. 35-53.

[10] J. NARBOUX. *A Graphical User Interface for Formal Proofs in Geometry*, in "the Journal of Automated Reasoning special issue on User Interface for Theorem Proving", to appear, 2006, http://www.lix.polytechnique.fr/Labo/Julien.Narboux/papers/JARuitp-narboux.ps.gz.

[11] F.-R. SINOT. *Call-by-Need in Token-Passing Nets*, in "Mathematical Structures in Computer Science", vol. 16, n^{o} 4, 2006.

[12] F.-R. SINOT. *Token-Passing Nets: Call-by-Need for Free*, in "Electronic Notes in Theoretical Computer Science", vol. 135, n^{o} 3, March 2006, p. 129–139.

Publications in Conferences and Workshops

[13] P. ARRIGHI, G. DOWEK. *Linear-algebraic lambda-calculus*, in "International workshop on quantum programming languages", P. SELINGER (editor). , Turku Centre for Computer Science General Publication, 33, 2006.

[14] J. CHRZASZCZ, J.-P. JOUANNAUD. *From OBJ to ML to Coq.*, in "Essays Dedicated to Joseph A. Goguen", K. FUTATSUGI, J.-P. JOUANNAUD, J. MESEGUER (editors). , Lecture Notes in Computer Science, vol. 4060, Springer, 2006, p. 216-234.

[15] G. DOWEK. *Truth values algebras and normalization*, to be published in Types 2006, 2006.

[16] B. GRÉGOIRE, L. THÉRY, B. WERNER. *A computational approach to Pocklington certificates in Type Theory*, in "FLOPS", M. HAGIYA, P. WADLER (editors). , Lecture Notes in Computer Science, Springer, 2006.

[17] B. GRÉGOIRE, L. THÉRY, B. WERNER. *A computational approach to Pocklington certificates in type theory*, in "FLOPS 2006", M. HAGIYA, P. WADLER (editors). , LNCS, vol. 3945, Springer, 2006.

[18] J.-P. JOUANNAUD. *Modular Church-Rosser Modulo.*, in "RTA", F. PFENNING (editor). , LNCS, vol. 4098, Springer, 2006, p. 96-107.

[19] J.-P. JOUANNAUD, A. RUBIO. *Higher-Order Orderings for Normal Rewriting.*, in "RTA", F. PFENNING (editor). , LNCS, vol. 4098, Springer, 2006, p. 387-399.

[20] F. KIRCHNER. *Store-based Operational Semantics*, in "Seizièmes Journées Francophones des Langages Applicatifs", INRIA, 2005.

[21] F. KIRCHNER. *A Finite First-Order Theory of Classes*, in "TYPES", Lecture Notes in Computer Science, Springer-Verlag, 2006.

[22] F. KIRCHNER, C. MUÑOZ. *PVS#: Streamlined Tacticals for PVS*, in "Proceedings of STRATEGIES'06", August 2006.

[23] F. KIRCHNER, F.-R. SINOT. *Rule-Based Operational Semantics for and Imperative Language.*, in "Proceedings of RULE'06", August 2006.

[24] J. NARBOUX. *Mechanical Theorem Proving in Tarski's geometry*, in "Proceedings of Automatic Deduction in Geometry 06", 2006.

[25] B. WERNER. *La Vérité et la Machine*, in "Images des Mathématiques – 2006", E. GHYS, J. ISTAS (editors). , Société Mathématique de France, 2006.

[26] B. WERNER. *On the strength of proof-irrelevant type theories*, in "Int. Joint Conf. Automated Reasoning — IJCAR 2006", U. FURBACH, N. SHANKAR (editors). , LNAI, vol. 4130, Springer, 2006.

[27] R. ZUMKELLER. *Formal Global Optimisation with Taylor Models*, in "Int. Joint Conf. Automated Reasoning — IJCAR 2006", U. FURBACH, N. SHANKAR (editors). , LNAI, vol. 4130, Springer, 2006.

Internal Reports

[28] P. ARRIGHI, G. DOWEK. *Linear-algebraic lambda-calculus: higher-order, encodings and confluence*, submitted for publication, Technical report, 2006.

[29] D. COUSINEAU, G. DOWEK. *Embedding Pure Type Systems in the lambda-Pi-calculus modulo*, submitted for publication, Technical report, 2006.

[30] G. DOWEK. *Gödel's system T as a precursor of modern type theory, Modern Type Theory*, manuscript, Technical report, 2006.

[31] G. DOWEK, O. HERMANT. *Two proofs of cut elimination for super-consistent theories*, in preparation, Technical report, 2006.

[32] C. MUÑOZ, G. DOWEK, C. PASAREANU. *A semantic Framework for plan execution languages*, submitted for publication, Technical report, 2006.

Miscellaneous

[33] D. COUSINEAU. *Un plongement conservatif des Systèmes de Types Purs dans le lambda-Pi-calcul modulo*, master thesis, University Paris VII, 2006.

[34] D. COUSINEAU, G. DOWEK. *Embedding Pure Type Systems in the lambda-Pi-calculus modulo*, submitted (TLCA 07), 2006.

[35] J. NARBOUX. *A formalization of diagrammatic proofs in abstract rewriting*, 2006, http://www.lix.polytechnique.fr/Labo/Julien.Na