INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team PROTHEO

Constraints, Mechanized Deduction and Proofs of Software Properties

Lorraine

THEME SYM

Table of contents

1. Team

PROTHEO *is a research project of* ***LORIA*** *(Research Laboratory in Computer Science and Control of Lorraine, UMR 7503), a laboratory shared by* ***CNRS*** *(National Center for Scientific Research),* ***INRIA*** *(National Institute for Research on Computer Science and Control),* ***UHP*** *(University Henri Poincaré Nancy 1),* ***Nancy2*** *(University Nancy 2) and* ***INPL*** *(National Engineering Institute of Lorraine).*

Christophe Ringeissen and Duc Tran moved to the Cassis team. Their activity is mainly described in the Cassis report.

Head of project

Claude Kirchner [DR INRIA]

Administrative assistant

Chantal Llorens [CNRS]

Research scientists

Frédéric Blanqui [CR INRIA]
Olivier Bournez [CR INRIA]
Isabelle Gnaedig-Antoine [CR INRIA]
Hélène Kirchner [DR CNRS, part time]
Pierre-Etienne Moreau [CR INRIA]
Christophe Ringeissen [CR INRIA, until 30/06/05]

Faculty members

Horatiu Cirstea [MC Nancy2]
Liliana Ibănescu [ATER until 31/08/05]

External collaborators

Luigi Liquori [CR INRIA, Sophia-Antipolis]

Technical staff

Laika Moussa [INRIA fixed-term engineer since 15/04/04]
Anne-Claire Lonchamp [INRIA fixed-term engineer since 15/12/04]
Yoann Toussaint [INRIA fixed-term engineer since 01/09/05]

Ph. D. students

Oana Andrei [INRIA since 01/10/05]
Emilie Balland [MENRT since 01/10/05]
Clara Bertolissi [MENRT since 01/10/02]
Guillaume Burel [ENS Lyon since 01/10/05]
Germain Faure [MENRT since 01/10/04]
Florent Garnier [INRIA since 01/11/03]
Emmanuel Hainry [MENRT since 01/09/04, with Calligramme]
Paulin Jacobé de Naurois [MENRT until 31/08/05, with Calligramme and Hong-Kong University]
Radu Kopetz [INRIA since 01/07/05]
Fabrice Nahon [High school teacher, since 01/09/02]
Antoine Reilles [CNRS since 01/10/03]
Colin Riba [MENRT since 01/10/04]
Anderson Santana [Brazil since 01/10/04]
Duc Tran [MENRT until 31/08/05]
Benjamin Wack [MENRT since 01/09/02]

Internships

Oana-Maria Andrei [Master, 15/04-15/09]
Emilie Balland [Master, 15/01-30/07]
Bai Bingyang [Master, 17/02-30/09]

Guillaume Burel [Master, 01/03-30/07]
André Chailloux [ENS Cachan, 30/05-08/07]
Daniel Graca [PhD Portugal, 01/01-30/06 and 01/10-31/12]
Clément Houtmann [ENS Cachan, 01/03-30/07, Master, 15/10-01/03/06]
Christophe Masson [IUT Nancy, 04/07-31/08]
Hejer Rejeb [ENSI Engineer School, Tunis, 14/02-13/06]
Zhen Wang [Hong-Kong University, 15/09-15/02/06]
Zhong Yiyang [Master, 24/02-30/07]

2. Overall Objectives

2.1. Overall Objectives

The PROTHEO project aims to design and implement tools for program specification, proof of properties and safe and efficient execution.

We are working on environments for prototyping such tools, on theorem provers specialized in proofs by induction and equational first-order proofs, on proof techniques involving constraints and rewrite rules. The project has three strongly connected research domains:

- Constraint solving,
- Mechanized deduction with rewrite rules and strategies,
- Theorem proving based on deduction modulo.

The team develops and maintains several software packages detailed later in this document. They allow us to test our ideas and results as well as to make them available to the community.

3. Scientific Foundations

3.1. Rewriting and strategies

Keywords: *functional programming*, *rewriting*, *rule based programming*, *strategies*.

Rewriting techniques have been developed since the 1970s and have been applied in particular to the prototyping of formal algebraic specifications and to the automated deduction of properties often related to program verification [78].

Rewriting techniques have been also used for describing inductive theorem provers, for verifying the completeness and coherence proofs for equational or conditional specifications, for defining first-order theorem provers, for solving equations in equational or conditional theories. Rewriting has been also applied to specific domains like, for example, the automatic demonstration of geometric properties or the verification of electronic circuits. This rewriting approach has proved extremely useful for simplifying search spaces, or for including decision procedures in general provers.

A common feature of (the evaluation of) functional languages and of theorem provers (including proof assistants) is the study of strategies. These strategies allow one, for instance, to guide computations and deductions by specifying which rule should be applied to which position in the term, or to restrict the search space by selecting only certain branches. In functional programming, we can also mention lazy evaluation and call-by-need strategy. In theorem proving, it is interesting to clearly separate inference rules and control strategies, since the correctness and completeness proofs are easier to obtain when using such an approach. Moreover, it is necessary to have a sufficiently expressive strategy language in order to express iteration, case reasoning, deterministic and nondeterministic choices. We have been studying strategies from the point of view of their specifications and their properties. We use them to formalize proofs in the demonstration and verification tools we develop.

Last but not least, rewriting is a fundamental paradigm for the description of transformations, either functional or not. Starting from our previous works on rewriting and strategies, we have introduced a new formalism generalizing λ-calculus and rewriting that we called ρ-calculus [5]. We have been studying the expressiveness of this general formalism and the properties of its various instances.

3.2. Constraints

Keywords: *combination problem, constraint solving, satisfiability.*

The notion of constraint has proved to be interesting in the modeling of various problems taken from a large variety of domains like mechanics, logic and management of human activities. The properties to satisfy are specified as a set of constraints for which it is important to determine if it admits a solution, or to compute a description of all solutions.

In the context of automated deduction, dealing with symbolic constraints on abstract domains like terms is of the greatest interest. For instance, syntactic unification is solving equational constraints over terms, and it is a fundamental notion for logic programming languages and automated theorem provers. The unification problem extends to the case of equational theories, where function symbols may admit some equational properties like the associativity and commutativity [68]. Other symbolic constraint systems may use predicates distinct from equality, like ordering constraints or membership constraints.

We are interested in the problem of combining symbolic constraint solvers for abstract (term-based) domains. We focus on the matching problem, which is the constraint solving process used when applying rewrite rules. The interest in matching is explained by its crucial role in the ρ-calculus, and more generally in rewrite engines.

3.3. Mechanized deduction

Keywords: *constraints, deduction, induction, paramodulation, resolution, rewriting.*

Developing methods and tools for verifying software is one of our main goals. To achieve it, we develop techniques and automated deduction systems based on rewriting and constraint solving.

Verifying specifications on recursive data structures often relies on inductive reasoning or equation handling, and uses operator properties like associativity or commutativity.

Rewriting, which enables us to simplify expressions and formulas, is now an essential tool for making the automated proof systems efficient. Moreover, a well founded rewriting relation can be used in a natural way to implement inductive reasoning. So we study termination of rewriting, as well as to guarantee termination of programs, in the scope of our study of rule-based programming languages, and to allow reasoning in automated deduction. A special effort is made for develop specific termination proof tools for rewriting strategies.

Constraints allow us to postpone complex symbolic problem solving,so that they can be solved in an expedient way. They also allow us to increase expressiveness of specification languages and to refine proof strategies.

Dealing with unification or orienting constraints with interpreted operators (like associative-commutative ones) gives the hope of obtaining much simpler automated proofs. Implementing these ideas has indeed allowed W. McCune [56], [72] to solve an open mathematical problem. Combining constraints and rewriting based simplifications induces complex problems, either theoretical as for example strategies completeness, or practical as for instance efficient implementation. We explore these techniques from these two point of views.

4. Application Domains

4.1. Application Domains

Keywords: *compilation, modeling, program transformation, proof of properties, prototyping, specification, verification.*

Our research applies to modeling, prototyping and verification of software components. To model these systems, we use rule based languages with constraints and strategies that allow one to quickly prototype applications. In particular the matching capabilities of such languages offer ease of expressivity for protocol specification and verification. We also intend to apply these techniques to model and check reactive as well as hybrid systems.

Constraint satisfiability, propagation and solving is of course in itself a main domain of application and has leaded to the creation of the Enginest Software company in 2000. Based on constraint solving, Plansuite, one of Enginest's products, is a global solution for transport and logistics planning. It allows users to anticipate, plan, manage and forecast the use of logistic resources.

The (safe) transformation of XML entities is mainly a rule based process. XSLT is one of the language used for that purpose. The combination of rewrite based transformations, strategies and typing is a natural application domain of the concepts and tools we are developing.

5. Software

5.1. Introduction

In this section, we only describe software that are distributed. Other software are developed within contracts and grants but they are not distributed yet (see Section 7).

5.2. TOM

Keywords: *compilation, pattern matching, rule based programming*.

Participants: Emilie Balland, Anne-Claire Lonchamp, Pierre-Etienne Moreau, Antoine Reilles, Yoann Toussaint.

Since 2002, we have developed a new system called *Tom*, presented in [76]. This system consists of a pattern matching compiler which is particularly well-suited for programming various transformations on trees/terms and XML documents. Its design follows our experiences on the efficient compilation of rule-based systems [8]. The main originality of this system is to be language and data-structure independent. This means that the *Tom* technology can be used in a C, C++ or Java environment. The tool can be seen as a Yacc-like compiler translating patterns into executable pattern matching automata. Similarly to Yacc, when a match is found, the corresponding semantic action (a sequence of instructions written in the chosen underlying language) is triggered and executed. *Tom* supports sophisticated matching theories such as associative matching modulo neutral element (also known as list-matching). This kind of matching theory is particularly well suited to perform list or XML based transformations for example. The main idea consists in encoding a DOM object into a term-based representation (a DOM NodeList becomes an associative list-operator), and then perform matching and subterm retrieving using the *Tom* pattern matching facilities. On the one hand, this approach is not comparable to XSLT. But, on the other side, the expressivity is very high since it is possible to combine powerful pattern matching constructs with the expressive power of Java.

Tom is documented, maintained and available at http://tom.loria.fr.

5.3. CARIBOO

Keywords: *induction, innermost outermost, local strategy, strategies, termination*.

Participants: Isabelle Gnaedig, Hélène Kirchner, Laika Moussa.

Since 2002, we have been developing a new proof tool called *CARIBOO* (for Computing AbstRaction for Induction Based termination prOOfs). Presented first in [64], and distributed since 2004, *CARIBOO* is a termination proof tool for rule-based programming languages, where a program is a rewrite system and query evaluation consists in rewriting a ground expression. It applies to languages such as ASF+SDF, Maude, Cafe-OBJ, or ELAN. *CARIBOO* is dedicated to termination proofs under specific reduction strategies, which

becomes of special interest when the computations diverge for standard rewriting. It deals in particular with the two basic strategies. The innermost strategy, specially useful when the rule-based formalism expresses functional programs, and central in their evaluation process, local strategies on operators, allowing to control evaluation strategies in a very precise way. The outermost strategy, useful to avoid evaluations known to be non terminating for the standard strategy, to make strategy computations shorter, and used for interpreters and compilers using call by name.

The proof technique of *CARIBOO* relies on an original generic inductive process, instantiated for the different considered strategies. For proving that a given term terminates, we proceed by explicit induction on the termination property, on ground terms with a noetherian ordering, assuming that any term less than the given term terminates (according to the given strategy). Proving termination on ground terms amounts to proving that the rewriting derivation tree starting from any ground term has only finite branches.

The derivation trees are simulated by proof trees starting from patterns built with a defined symbol and variables. They are developed by alternatively using two main concepts, namely narrowing and abstraction, whose definition depends on the considered rewriting strategy. More precisely, narrowing schematizes all rewriting possibilities of the terms in the derivations. The abstraction process simulates the normalization of subterms in the derivations, according to the strategy, by replacing them by special variables, denoting any of their normal forms. By abstraction, induction elegantly allows to simulate normalization of subterms, without to effectively get the normal forms. The proof procedure terminates with success when finiteness has been inductively established for each proof tree.

The induction ordering is specified in a constructive way: it is not given a priori but the proof process generates ordering and abstraction constraints, that are only required to be satisfiable. These constraints are automatically solved, or delegated to an external constraint solver, or to the user.

This year, a new release has been developed, improving the installing procedure of the software, the functionalities of the specification editor, and the narrowing process in the proof procedure.

CARIBOO is documented, maintained and available at http://protheo.loria.fr/softwares/cariboo.

5.4. COLOR

Keywords: *Coq*, *proof*, *rewriting*, *termination*.

Participant: Frédéric Blanqui.

CoLoR is a Coq [57] library on rewriting and termination. It is intended to serve as a basis for certifying the output of termination checkers like Cariboo, AProVE, Torpa,.... It has its origin in Sébastien Hinderer's formalization in Coq of polynomial interpretations [66]. It has then been extended with dependency pairs and arguments filterings. Adam Koprowski (Eindhoven) added the multiset ordering extension [70]. Solange Coupet-Grimal and William Delobel (Marseille) added the multiset path ordering (MPO) [58].

As a by-product, *CoLoR* provides general purpose libraries on vectors, integer polynomials with multiple variables, finite multisets and first-order terms.

CoLoR is distributed under CeCILL license on http://color.loria.fr/.

5.5. MHTML

Keywords: *generic modularity*, *xhtml*.

Participants: Claude Kirchner, Hélène Kirchner, Anderson Santana, Christophe Masson.

mHTML is a simple implementation of generic parametricity for HTML. It allows for the declaration of modules and of their use in import or parametrisation constructs. It is available at url http://mhtml.loria.fr.

5.6. ELAN

Keywords: *computation*, *deduction*, *rules*, *specification*, *strategies*.

Participants: Claude Kirchner, Pierre-Étienne Moreau.

The *ELAN* system provides an environment for specifying and prototyping deduction systems in a language based on rewrite rules controlled by strategies. It offers a natural and simple logical framework for the combination of computation and deduction paradigms as it is backed up by the concepts of ρ-calculus and rewriting logic. It supports the design of theorem provers, logic programming languages, constraint solvers and decision procedures and offers a modular framework for studying their combination.

ELAN was developed until 2003. It is still documented, maintained and available at http://elan.loria.fr.

6. New Results

6.1. Rewriting

Keywords: *conditional rewriting*, *rewriting*, *rewriting calculus*, *strategies*, *types*.

We have developed the foundational study of the rewriting calculus, studied the combination of conditional rewriting with lambda-calculus and continued to study probabilistic rewrite systems.

6.1.1. Explicit ρ-calculus

Participants: Horatiu Cirstea, Germain Faure, Claude Kirchner.

Theoretical presentations of the ρ-calculus often treat matching constraint computations as an atomic operation (although matching constraints are explicitly expressed). Concrete implementations have to take a much more realistic view: computations needed to find the solutions of a matching equation (in some matching theories) can have a strong impact on the properties of the calculus. Moreover, the application of the obtained substitutions usually involves term traversals.

Following the works on explicit substitutions for λ-calculus, we proposed, studied and exemplified [52] a ρ-calculus that handles explicitly the resolution of the matching constraints and the application of the obtained substitutions. We have also shown that the approach is modular and we have introduced a calculus handling explicitly only the substitution application and another one where the matching constraints are solved at the object level while the resulting substitutions are applied at the meta-level [20]. All these calculi can be extended to arbitrary matching theories.

The explicit substitution application studied initially [52] is not optimal since the possible complexity of term traversals is not taken into account. We have thus composed and improved the previous explicit versions and we introduced a calculus that offers support for the composition of substitutions [20]. We proved the confluence of the calculus and the termination of the explicit constraint handling part.

Moreover, in this approach the matching constraints with no solution can be eliminated earlier in the reduction process leading to a more efficient evaluation. This can be achieved by integrating in the explicit calculus the approach already used for the plain calculus [55].

6.1.2. Graphs in the ρ-calculus

Participants: Clara Bertolissi, Horatiu Cirstea, Claude Kirchner.

Starting from the classical untyped ρ-calculus and in collaboration with Paolo Baldan from the University of Venice, we have proposed an extension of the calculus, called *graph rewriting calculus*, handling structures containing sharing and cycles, rather than simple terms [45].

The classical ρ-calculus is naturally generalized by considering lists of constraints containing unification constraints in addition to the standard matching constraints. The evaluation rules are adapted to the new syntax leading to an enhanced expressive power for the calculus. In this new formalism we can represent and manipulate elaborated objects like, for example, regular infinite entities.

Several aspects of the calculus have been investigated so far, like its properties and its relationship with other existing frameworks [11]. More precisely we have shown that the calculus is confluent over equivalence classes of terms, under some linearity restrictions on patterns [36].

A natural but important result we achieved is the conservativity of the graph rewriting calculus versus the two formalisms that inspired its design, that is the classical ρ-calculus and a version of the λ-calculus extended with a `letrec` like construct, used for modelling explicit recursion.

Finally, the possibility of representing structures with cycles and sharing naturally made us compare the graph rewriting calculus and first-order term graph rewriting. We have thus proposed an encoding of term graphs and a simulation of their reductions in the graph rewriting calculus [23].

6.1.3. *Typed programming with ρ-calculus*

Participants: Horatiu Cirstea, Germain Faure, Claude Kirchner, Luigi Liquori, Benjamin Wack.

We have extensively studied a second-order typed ρ-calculus *à la* Curry called $\rho^{\ll}_{\forall}$, whose type system is mainly inspired by polymorphic typed λ-calculus *à la Leivant*. It had been proved before [71] that such a typed calculus enjoys subject reduction. However, as a wanted feature it does not enforce termination of typed terms: we have shown the encoding of some interesting terms leading to infinite reductions by the use of pattern matching features of the calculus.

More recently, we have studied type inference for this typed calculus [30]. We have provided an inference algorithm in the fashion of Damas and Milner's *W* algorithm, which is used in most of the functional languages derived from *ML*. The ability to automatically type-check terms, together with the encoding of term rewriting systems and strategies we have provided in earlier works, gives a full process for ensuring the safety of a program described by rewrite rules.

The consequence is that our typing discipline fits for a programming language since we are interested in type consistency and in recursive (potentially non-terminating) programs. Conversely, it is not adapted for defining a logical framework, since normalization is strongly linked to consistency, so it definitively differs from other type systems we have proposed for the ρ-calculus [1].

Since the rewriting calculus can be used for the encoding of term rewrite systems and since there is a strong relathionship between these systems and the modern functional languages, we proposed [29] the rewriting calculus as an intermediate language in the implementation of functional languages with pattern-matching features. We have defined an interaction net encoding of the rewriting calculus terms arising from the compilation, where pattern-matching and traditional reduction can proceed in parallel without additional overheads.

As we have mentioned above, the automatic encoding we have proposed can be done only for a restricted class of rewrite systems and only some classical reduction strategies (*e.g.* outermost) have been encoded. Therefore, we currently investigate the way this can be generalized to unrestricted rewrite systems and to a larger class of reduction strategies.

6.1.4. *Typed ρ-calculi for logics*

Participants: Horatiu Cirstea, Claude Kirchner, Luigi Liquori, Benjamin Wack.

We are also interested in more elaborated type systems and we have proposed a ρ-cube which generalizes Barendregt's λ-cube. First, with Gilles Barthe and Luigi Liquori [1], we have proposed and studied a framework called Pure Pattern Type Systems (P^2TS), an algebraic extension of Pure Type Systems. In P^2TS, rewrite rules can be considered as lambda terms with patterns, and the application of a rule is the application of λ-calculus, enhanced with matching capabilities. This context uniformly unifies higher-order functions and rewriting mechanisms, together with a powerful type system. Thus, it is particularly adapted for formalizing the foundations of programming (meta)languages and proof assistants. We have proved various standard properties such as subject reduction, confluence and strong normalization [79].

More recently, we have explored the applications of this typed calculus in logics [12]. Namely, we have shown how to use a subset of P^2TS as a proof-term language for natural deduction modulo, extending the Curry-Howard-de Bruijn isomorphism for this class of logical formalisms. The pattern matching featured in the calculus allows us to model any congruence given by a term rewriting system.

We have characterized how proofs can be denoted by P^2TS terms and we have studied the interest of our proof-term language for the issue of cut elimination. Finally, we have explored some relations between our

proof-term language and other formalisms: extraction of λ-terms and/or rewrite rules from P^2TS-terms, but also automated generation of proof-terms by a rewriting-based language.

6.1.5. Models of the ρ-calculus

Participant: Germain Faure.

Up to now, the ρ-calculus has been studied only from the perspective of its operational semantics. In collaboration with Alexandre Miquel, we have proposed a first attempt to a Scott-style semantics for the ρ-calculus, when restricted to ML-style pattern-matching.

We define a notion of ρ-model in the category of Scott-domains, and prove the soundness of the reduction rules w.r.t. this notion of ρ-model. We then show that any universal domain $D = (D \to D)$ with a top element (including Scott's historical D_∞) can be given the structure of a ρ-model by interpreting the structure construction of the ρ-calculus by the binary sup. From this, we deduce that the full ACI-theory of the ρ-calculus is conservative w.r.t. the $\beta\eta$-theory for normalizable λ-terms.

The link between monads is strongly examined and then we go a step further by proposing an abstract definition of models for the ρ-calculus in terms of categories. The computational ingredients of the calculus are thus clearly identified. Also, these results lead to the first intuitions for a separation theorem for the ρ-calculus.

6.1.6. Higher-order conditional rewriting

Participants: Frédéric Blanqui, Claude Kirchner, Colin Riba.

We extended to conditional rewriting the work of Dougherty [62] and Müller [77] on the combination of λ-calculus and algebraic rewriting. We considered the following two cases: when β-reduction is used or not in the evaluation of conditions. Incidentally, we improved Dougherty's result by requiring weak normalization of β-reduction (*i.e.* existence of a terminating β-reduction) instead of strong normalization (*i.e.* termination of every β-reduction). Finally, we provided a syntactic condition for ensuring orthogonality, hence confluence, in case of conditions with higher-order terms. This is the first study on the confluence of λ-calculus with higher-order conditional rewriting [38].

6.1.7. Rewriting and probabilities

Participants: Olivier Bournez, Florent Garnier, Claude Kirchner.

In [50], we presented an abstract way to implement probabilistic strategies in rule based languages. This year, probabilistic strategy already coded in ELAN4 have been ported in the TOM language. A CSMA-CA simulator has been implanted in TOM too.

We also started to work on techniques for proving almost sure termination of a set of rewrite rules, in particular on techniques that ensure that the mean time before termination is always finite. These techniques have been presented in [26].

In additionm we found sufficient conditions entailing a specific kind of combination between Probabilistic Rewrite Systems and Term Rewrite Systems to terminate in a finite mean time. Thanks to this result, we provided a proof of the termination of the simulation algorithm of the CSMA/CA protocol in a finite mean time.

Our more recent work deals with sufficient criteria entailing the termination in a finite mean time of the combination of two Probabilistic Rewrite Systems.

6.2. Rule based programming

Keywords: *abstract data-types, algebraic specification, compilation, pattern matching.*

We are studying the design and the implementation of rule based programming languages. We are interested in modularizing our technologies in order to make them available as separate elementary tools.

Also, we continue our study on the application of rule based languages. In particular, we study their applications to XML transformations as well as the analysis and the certification of program transformations.

6.2.1. Compilation and optimization of pattern-matching

Participants: Emilie Balland, Pierre-Etienne Moreau, Antoine Reilles.

In collaboration with Christophe Ringeissen and Marian Vittek, we have designed a new formalism, called *Tom*, which allows us to integrate equational pattern matching facilities into imperative languages such as C, Java or Caml. This new formalism has been presented in [76].

Tom is a language extension which adds new matching primitives to an existing imperative language. From an implementation point of view, it is a compiler which accepts different native languages: C, Java, and Caml. The compilation process consists in translating these new matching constructs into the underlying host language. An important feature of *Tom* is to support equational matching. In particular, list matching, also known as associative matching with neutral element. The compilation process is performed in two steps: patterns are compiled into an intermediate language, which is later translated into C, Java, or Caml. Recently, we have introduced a program transformation phase [37] which optimizes the intermediate program in order to improve its efficiency. As a result, *Tom* generates an efficient implementation of pattern matching, comparable to very optimized implementations such as those in *ELAN* or *ASF+SDF*.

6.2.2. Certification of pattern-matching compilation

Participants: Claude Kirchner, Pierre-Etienne Moreau, Antoine Reilles.

We are studying the certification of pattern-matching compilation for *Tom* [34]. By giving a certification of the fact that the *Tom* compiler produces correct target code for the *Tom* specification, we improve our confidence in the *Tom* compiler and the rule based approach. By giving, for each execution of the compiler a certification that the produced code meets the specification (or a failure), we can have much more information for debugging the compiler. It is particularly important in the case of associative matching for example.

Our first work has been to formalize the relations between the algebraic terms and the concrete objects representing them. Since *Tom* is data-structure independent, we needed to have a formal view of the mapping mechanism used in *Tom*, defining a general framework allowing to anchor algebraic pattern-matching capabilities in existing languages like C, Java or ML. This provides a specific instance of what we call a formal island. Then, using a powerful enough intermediate language, which could be viewed as a subset of $C \cap Java \cap ML$, we formalize the behavior of compiled code and define the correctness of compiled code with respect to pattern-matching behavior. This allows us to prove the equivalence of compiled code correctness with a generic first-order proposition whose proof could be achieved via a proof assistant or an automated theorem prover. We worked with Damien Doligiez from the Cristal project on how to connect our tool with the Zenon theorem prover [61], and how to help the theorem prover proving these goals. We then extended these results to the multi-match situation characteristic of the ML like languages. The approach has been implemented on top of the *Tom* compiler, and used to validate some syntactic matching construction. This prototype implementation will be extended to support full multi-match constructions and to automatize the cooperation with Zenon.

The extension of this method to matching with theories, like associative matching or AC-matching is a challenging task, as many new problems are still to be solved, like proving that the matching-code enumerates all possible matching solutions and formalizing these enumerations.

6.2.3. Generic modularity

Participants: Claude Kirchner, Hélène Kirchner, Anderson Santana.

Formal islands is a generic approach allowing to anchor in existing languages formal constructions. In this framework currently under development in the team, we study the anchoring of modularity capabilities like importations and parametrisation. To exemplify the approach and its capabilities, we have developed a specific instance for HTML, a widely used but unmodular language. The language design is presented in [33] and implemented in a distributed prototype called mHTML.

6.2.4. Rewriting, constraints and business rules

Participants: Horatiu Cirstea, Claude Kirchner, Radu Kopetz, Pierre-Etienne Moreau.

Most business rule systems, and the Ilog Rule system in particular, use the RETE algorithm to discriminate the rules which have to be applied and fired. In the project, we have developed several compilation algorithms based on the syntactic structure of the rules (e.g. [8]).

We have proposed a new formalization of the RETE algorithm [53] as well as a first formalization of the relationship between these two approaches: the classical term-based indexing algorithm and the RETE based production systems [54].

We recently introduced the notion of negative-pattern: they denote a set of terms which does not contain some given values. We started to develop a formal basis as well as a matching algorithm, based on constraint solving. The integration pattern matching and constraint solving should become the theoretical foundation of a more expressive rule based language where constraints could be used to improve the expressivity of the patterns and the actions used in business rules.

6.2.5. Types for XML transformation rules

Participants: Horatiu Cirstea, Claude Kirchner, Luigi Liquori, Pierre-Etienne Moreau, Anderson Santana, Benjamin Wack, Zhen Wang.

We have contributed to the survey and analysis of the relevant existing works on typing of rules, in particular on typing of constraint logic programs and discuss applicability of these approaches to the reasoning and query languages under development in the REWERSE network of excellence such as XPathLog and Xcerpt [51]. Starting from this study we have proposed [40] a type system for a subset of the languages developed and studied in the context of the REWERSE network of excellence. We study two approaches to such a type system, which are based on descriptive and prescriptive typing. As an example rule language we use the XML query language Xcerpt and we intend to investigate *Tom*.

6.2.6. Rules and strategies for generation of chemical reactions

Participants: Oana-Maria Andrei, Olivier Bournez, Liliana Ibănescu, Hélène Kirchner.

Rule-based programming has been recently applied to the field of automated generation of chemical reaction mechanisms. We explored in [27] a class of graphs and a graph rewriting relation where vertices are preserved and only edges are changed. We have shown how to represent cyclic labeled graphs by decorated labeled trees or forests, then how to transform trees into terms. A graph rewriting relation has been defined, then simulated by a tree rewriting relation, which can be in turn simulated by a rewriting relation on equivalence classes of terms. As a consequence, this kind of graph rewriting can be implemented using term rewriting.

This study is motivated by the design of the *GasEl* system for the generation of kinetics reactions mechanisms. In *GasEl*, chemical reactions correspond to graph rewrite rules and are implemented by conditional rewriting rules in *ELAN*. The control of their application is done through the *ELAN* strategy language.

6.2.7. Implementing deep inference in Tom

Participants: Horatiu Cirstea, Pierre-Etienne Moreau, Antoine Reilles.

In collaboration with Ozan Kahramanogullari from Dresden university, we worked on the calculus of structures and proposed a flexible implementation in *Tom*. This allowed to perform various kind of experimentations to improve the implementation and find a good proof search strategy. This also lead to a new version of the calculus providing better computational properties, and reducing the search space for proof search. The BV system was modified to reduce the non-determinism of the calculus, leading to a more complex calculus, with conditional rules. This calculus has been proven correct and equivalent to the original BV system, and implemented in *Tom* [32]. The strategy to explore the search space was described using the imperative features of the language, while the rules were specified using equational pattern matching. This method allowed to implement a *global search* strategy which combines the advantages of both depth- and breadth-first search strategies, and allows use of the Java language to describe variations in the strategy without losing completeness of the proof search, where higher level languages do not provide this flexibility.

6.2.8. Representation of abstract data-types

Participants: Pierre-Etienne Moreau, Antoine Reilles.

In collaboration with Mark van den Brand and Jurgen Vinju of CWI at Amsterdam, we have presented a Java back-end for ApiGen [22], a new tool that generates implementations of abstract syntax trees. The generated code is characterized by strong typing combined with a generic interface and maximal subterm sharing for memory efficiency and fast equality checking. The goal of this tool is to obtain safe and efficient programming interfaces for abstract syntax trees.

The contribution of this work is the combination of generating a strongly typed data-structure with maximal subterm sharing in Java. Practical experience shows that this approach can be used not only for tools that are otherwise manually constructed, but also for internal data-structures in generated tools.

6.2.9. Quotient types in functional programming

Participant: Frédéric Blanqui.

In the Modulogic project of the ACI Sécurité Informatique, we are developing with Pierre Weis (project Cristal, INRIA Rocquencourt), and Thérèse Hardin (LIP6, Paris) for the theoretical side, a program generation tool that produces construction functions for quotient types in OCaml. Quotient types are types whose values represent equivalence classes modulo some equational theory. Invariants like "the list is sorted", "there is no element occuring twice", ...can be seen as quotient types modulo common algebraic equational theories like commutativity, idempotence, ...Generating construction functions for ensuring such invariants may be difficult. Consider for instance the combination of associativity (for having right combs) and idempotence for a binary constructor C. Building the representive of the equivalence class of $C(x, y)$ when both x and y are the representatives of their equivalence class, requires to test a number of patterns proportional to the size of y. For instance, $C(x, C(y, C(x, y)))$ must be represented by $C(x, y)$. We are already able to generate construction functions for some combinations of such theories. In addition, we provide an option to generate code that ensures that representatives of quotient types are indeed maximally shared. Providing a truly general tool requires further research. In particular, the class of acceptable rewrite rules and equational theories is still unclear.

6.3. Mechanized deduction

Keywords: *completion procedures*, *constrained theories*, *decision procedures*, *deduction modulo*, *equational proofs*, *induction proofs*, *termination*.

We continued to develop both the inductive and size-based approaches to termination of rule-based programs. We also began to develop a Coq library for certifying termination proofs. Finally, we developed the use of "good proofs" in automated deduction.

6.3.1. Inductive strong termination proofs

Participants: Isabelle Gnaedig, Hélène Kirchner.

In previous works, we had proposed a method for specifically proving, by explicit induction, termination of rewriting under strategies. The proof principle consists, for a given term rewriting system, in establishing on the ground term algebra that every term terminates i.e. has only finite derivations, assuming that any smaller term terminates. For that, we develop proof trees representing the possible derivations from any term using the term rewriting system. Two mechanisms are used, namely abstraction, introducing variables that represent ground terms in normal form and schematize normalization of subterms, and narrowing, schematizing rewriting in different ways according to the ground instances it represents. These two mechanisms deal with constraints used to control the growth of the proof trees.

Although they are based on the same above general principle, we gave three distinct procedures, respectively for proving termination of rewriting with the innermost, the outermost and the local strategies. The number, the definition of inference rules, as well as the strategy to apply them and the constraints generated by the proof process have been designed specifically for each of the three strategies.

This year, we have worked on a factorization of these three procedures, in order to give a unified view of our proof induction principle for termination with strategies. For that, we have precisely identified their common

mechanisms on one hand, and the features specific to the given strategies on the other hand. We then have obtained a generic algorithm, combining three proof steps corresponding exactly to the basic principles of the induction mechanism: an abstraction, a narrowing and a stop step. Moreover, these steps are combined in a very simple way with the usual combinators “try” and “repeat”. We then have proved the suprising result that the steps are applied in the same order to simulate the rewriting relation, whatever the rewriting strategy. This was not the case for the previous distinct procedures. Instantiating the generic algorithm to obtain a specific termination proof procedure for one of the three above strategies then consists in instantiating the proof steps into inference rules, and the the combinator “try” into either a “skip” or a “stop” when trying a rule does not succeed.

A generalized definition of constraints has also been given, as well as a common principle to relax the induction hypothesis. Technical details vary following the three strategies. Their specific instances are managed by the specific inference rules. This work has been submitted to a journal [41], and a short version presented to the APPSEM II workshop [31].

6.3.2. Weak termination

Participants: Isabelle Gnaedig, Hejer Rejeb.

Last year, we adapted our inductive termination proof principle to the case of weak termination [65]. As it is lying on proof trees representing rewriting trees, our proof process is also well suited for the weak termination proof. Indeed, weak termination requires that at least one computation branch starting from each term is finite. So we have adapted our method to just develop branches of the proof trees, that represent one terminating branch of the rewriting tree of any ground term. We proceed by developing the trees in a breadth-first manner, and by cutting redundant branches, that represent in fact different ways of rewriting the same ground terms. Redundancy is detected in comparying the most general unifiers of a same narrowing step in the proof process.

This weak termination proof procedure has been studied this year with a view to an implementation in TOUNDRA, the prototype of toolbox for verification and proofs of rule-based program properties, developed last year, already integrating CARIBOO and a completeness verification procedure. An algorithmic translation of the abstract formalism of the weak termination procedure has first been proposed. The different mechanisms have been decomposed in a maximal way, to give basic modules that can be used independently.

Then, two of these modules, a matching algorithm and a unification algorithm, chosen for their usefulness in the context of rule-based programming, were implemented in TOM and integrated in TOUNDRA, in order to be at the same time available as independent tools, and useful for the weak termination proof procedure [42]. This work has been realized in the context of the TOUNDRA project of the PRST-IL theme on Quality and Safety of Software of the CPER 2000-2006.

6.3.3. Size-based termination

Participants: Frédéric Blanqui, Claude Kirchner, Colin Riba.

Last year, we generalized to rewriting and dependent types various works on ML-like function definitions based on the way inductive types are usually constructed: as fixpoints of monotone operators on the set of terms [46]. The number of steps necessary for reaching a term in such a fixpoint gives a simple and natural measure, the “size”, which is finer than the syntactic structure. By enriching the syntax of types with size annotations, we can express that a term has a size (strictly) smaller than another term. In [24], we proved that this termination criterion is decidable by reducing the problem of type-checking with size annotations to the problem of solving ordering constraints on size variables, and then by reducing this problem to linear programming. An interesting point is that, with dependent types, completeness requires the existence of a most general solution. We are now investigating how to extend this work with more complex size annotations in order, for instance, to be able to automatically prove that some sorting functions preserve the size of a list.

6.3.4. Certification of termination proofs

Participant: Frédéric Blanqui.

After Sébastien Hinderer formalized in Coq the termination criterion for rewrite systems based on polynomial interpretations [66], we decided to develop a Coq Library on Rewriting and termination *CoLoR* (see http://color.loria.fr/) in order to be able to certify termination proofs. In March, we released a first version of CoLoR after having formalized dependency pairs. Since then, the library has been extended with arguments filterings by myself, multisets by Adam Koprowski (Eindhoven) [70] and MPO by Solange Coupet-Grimal and William Delobel (Marseille). CoLoR is distributed under CeCILL license. We also discussed an XML format for termination proofs with some people participating to the annual termination competition (see http://www.lri.fr/~marche/termination-competition/).

6.3.5. Abstract canonical presentations

Participants: Claude Kirchner, Guillaume Burel.

Solving goals, like deciding word problems or resolving constraints, is much easier in some theory presentations than in others. What have been called "completion processes", particularly in the study of equational logic, involves finding appropriate presentations of a given theory to solve easily a given class of problems.

We have designed a general proof-theoretic setting within which completion-like processes can be modeled and studied. This framework centers around well-founded orderings of proofs. It allows for abstract definitions and very general characterizations of saturation processes and redundancy criteria [60].

In his master's thesis [39], Guillaume Burel gave complete proofs that this framework can be instantiate to ground and standard [69] completion and to natural deduction. For the first proof systems, we had to compare proof representations in equational logic, namely proof terms like in the rewriting logic [73], or proof by replacement of equal by equal, to choose the most adapted one. For the second one, a generalization of the framework was needed. A conservative one was given, in the sense that all results of the original framework still hold [28].

6.3.6. Induction and rewriting

Participants: Claude Kirchner, Hélène Kirchner, Fabrice Nahon.

Following our work with Eric Deplagne on a proof theoretic framework to perform rewrite based inductive reasonning [59], we have developed proof search inference systems based on narrowing. This approach does not need to compute induction variables nor induction schema which are automatically infered by the narrowing process. Soundness and completeness of the system is proved and the system examplified on several cases [43].

6.3.7. Superposition-based satisfiability procedures and their combination

Participants: Hélène Kirchner, Christophe Ringeissen, Duc Tran.

In collaboration with Silvio Ranise, Christophe Ringeissen et Duc Khanh Tran from the Cassis project, we have studied how to efficiently combine satisfiability procedures built by using a superposition calculus with satisfiability procedures for theories, for which the superposition calculus may not apply (e.g., for various decidable fragments of Arithmetic). Our starting point is the Nelson-Oppen combination method, where satisfiability procedures cooperate by exchanging entailed (disjunction of) equalities between variables. We have shown that the superposition calculus deduces sufficiently many such equalities for convex theories (e.g., the theory of equality and the theory of lists) and disjunction of equalities for non-convex theories (e.g., the theory of arrays) to guarantee the completeness of the combination method. Experimental results on proof obligations extracted from the certification of auto-generated aerospace software confirm the efficiency of the approach. Finally, we have shown how to make satisfiability procedures built by superposition both incremental and resettable by using a hierarchic variant of the Nelson-Oppen method [35].

6.4. Computability and complexity

Keywords: *analog computations*, *complexity*, *computability*, *implicit complexity*.

We focus on computational models working over reals with a discrete and continuous time. Concerning discrete time models, we mainly focus on the model of computation introduced by Blum, Shub and Smale over the reals, later on extended over arbitrary structures by Poizat. We have obtained several syntactic characterizations of complexity classes in this model. These characterizations subsume classical ones when restricted to booleans. Concerning continuous time models, we have provided machine independent characterization of elementary computable functions over reals. We have discussed the computational power of several models.

6.4.1. Complexity in the Blum, Shub, Smale model of computation

Participants: Olivier Bournez, Paulin Jacobé de Naurois.

In order to model discrete time computation over real numbers, Blum, Shub and Smale have introduced in 1989 a new model of computation, referred as BSS machine or sometimes real Turing machine. In this model, the complexity of a computational problem is given by the number of elementary arithmetical operations and comparisons needed to solve this problem, independently of the underlying representation of real numbers. This makes it very different from the recursive analysis model, and occurs to be a very natural way to describe computational problems over real numbers. The BSS model of computation has been later on extended to the notion of computation over arbitrary logical structure. Since classical complexity theory occurs to be the restriction of this general model to boolean structures, it gives a new insight on previous questions on classical complexity theory and its links with logic.

We provided several machine-independent characterizations of some complexity classes, over an arbitrary structure, in this model.

More specifically, in [17], we provided a characterization of partial recursive functions over any arbitrary structure. We show that polynomial time over an arbitrary structure can be characterized in terms of safe recursion. We show that polynomial parallel time over an arbitrary structure can be characterized in terms of safe recursion with substitutions.

In [16], we showed that the levels of the polynomial hierarchy correspond to safe recursion with predicative minimization and the levels of the digital polynomial hierarchy to safe recursion with digital predicative minimization. Also, we show that polynomial alternating time corresponds to safe recursion with predicative substitutions and that digital polynomial alternating time corresponds to safe recursion with digital predicative substitutions.

In [25], we proposed a modification of the logical characterizations of $P_{\mathcal{K}}$ and $NP_{\mathcal{K}}$ given by Grädel and Gurevich in order to make them applicable to any computable structure $\mathcal{K}$.

6.4.2. Analog computations

Participants: Olivier Bournez, Daniel Graça, Emmanuel Hainry.

There are many ways to model analog computers. Unifying those models is therefore a crucial matter as opposed to the discrete case, as there is no property stating that those models are equivalent. It is possible to distinguish two groups in these models: on one side, continuous time models; on the other side, discrete time models working on continuous structures as a model derived from Turing machines. The first group contains in particular some sets of functions defined by Moore in [74]. The main representative of the second group are the real computable functions and a subclass of this set: the set of elementary computable functions.

There are few comparisons between classes of functions from the first group and from the second group, and in particular, there were almost no result of equality.

In [18], we presented an *analog* and *machine-independent* algebraic characterization of elementarily computable functions over the real numbers in the sense of recursive analysis: we prove that they correspond to the smallest class of functions that contains some basic functions, and closed by composition, linear integration, and a simple limit schema.

We generalized this result to all higher levels of the Grzegorczyk Hierarchy.

Concerning *recursive analysis*, our results provide machine-independent characterizations of natural classes of computable functions over the real numbers, allowing to define these classes without usual considerations

on higher-order (type 2) Turing machines. Concerning *analog models*, our results provide a characterization of the power of a natural class of analog models over the real numbers.

In [4], we presented an *analog* and *machine-independent* algebraic characterization of (all, i.e. non-necessarily elementarily) computable functions over the real numbers that preserve integers: we prove that they correspond to the smallest class of real functions containing some basic functions and closed by composition, linear integration and a very natural unique minimization schema.

The approaches from [49] on one side, and [3], [48] and [18] on the other side, were mixed together and generalized in [19] to obtain characterizations of elementarily computable functions, classes from Grzegorczyk's Hierarchy, and recursive functions in terms of the most natural and robust continuous operators.

In some sense, the problem we solve there is the definition of a minimization operator, which is strong enough to get at least Turing machine power, but not too strong to get the technical problems of [75], nor non-robust super-Turing Zeno phenomena of [44], [47], [63], [67], [75].

In [15], we compared the computational power of several models of continuous time dynamical systems with the power of classical Turing machines. We surveyed several results and presented some new ones about the power of several classes of smooth systems.

Recently, in collaboration with Manuel Campagnolo, we worked on relating recursive analysis with another model of analog computation: Shannon's General Purpose Analog Computer (GPAC). This model was shown to be less powerfull than recursive analysis, but this result relies on restrictive definitions of the GPAC, and we showed that relaxing those definitions, it is possible to compute the functions computable in the sense of recursive analysis.

7. Contracts and Grants with Industry

7.1. Averroes

Participants: Frédéric Blanqui, Olivier Bournez, Horatiu Cirstea, Claude Kirchner.

The main goal of the RNTL Averroes project (Analysis and VERification for the Reliability Of Embedded Systems) is to contribute to the development of formal methods able to verify multi-form (quantitative) functional and non-functional properties, that appear in industrial problematics. The participants of this project are France Télécom R&D, CRIL Technology Systèmes Avancés, the LaBRI in Bordeaux, the LORIA in Nancy, the PCRI in Saclay and the LSV in Cachan.

The project relies on results obtained in previous National Network for Research on Telecommunication (RNRT) project Calife. It aims at consolidating some of them (e.g. implementation of theoretical results in proof tools) or at generalizing the considered framework in order to solve problems that appeared in practice. For instance, the consideration of stochastic systems, or of properties dealing with consumption of resources. A description of the project can be found at http://www-verimag.imag.fr/AVERROES/.

We are particularly involved in Lot 2 ("applications"), concerning applications, in Lot 3 ("tests and animations"), concerning the graphical animation of execution traces in the platform developed by the project, in Lot 4 ("model technologies"), concerning modeling technologies for probabilistic and stochastic systems, and for guaranteeing complexity bounds on consumption of resources, and in Lot 5 ("verification technologies"), about adding rewriting to *Coq*, and about the mechanization of deduction.

7.2. Manifico

Participants: Horatiu Cirstea, Claude Kirchner, Radu Kopetz, Anne-Claire Lonchamp, Pierre-Etienne Moreau.

The main goal of the Manifico project is to improve the expressivity, the efficiency and the modularity of rule based systems. In particular, we are interested in studying how the use of constraints can improve the expressivity of rule based languages. The Protheo and the Contraintes INRIA teams are involved in this project. **ILOG** is the industrial partner.

By studying and understanding the relationship between pattern matching, RETE algorithm and constraint solving, one of our goals is to develop some new compilation algorithms which can combine the best of these three technologies.

8. Other Grants and Activities

8.1. Glossary

AICA Associazione Italiana per l'Inforatica ed il Calcolo Automatico

ANR National Agency for Research

ARA ANR Fundamental Research Action

ARASSIA ARA on Security, embedded Systems and Ambient Intelligence

ACI FNS Concerted and Incentive Action

ACISI ACI on Computer and Software Security

CISSI Comité interministériel pour la sécurité des systèmes d'information

Compulog ESPRIT network on Computational Logic

CSD Conseil Scientifique de la Défense

CPER Planning Contract between the Government and the Region

ERCIM European Research Consortium for Informatics and Mathematics

ESPRIT Information Technologies Program of the European Union

FNS National Fund for Science

GDR CNRS Research Group

GDR ALP GDR on Algorithmics, Languages and Programming

PAI Integrated Action Programme

PRST CPER Pole of Scientific and Technological Research

PRST-IL PRST devoted to Software Intelligence

QSL PRST-IL project on Quality and Safety of Software

RNTL National Network on software Technology

8.2. Regional initiatives

On the level of the CPER 2000-2006, we are involved in the PRST on Software Intelligence coordinated by Hélène Kirchner. PROTHEO is involved in the PRST-IL theme on Quality and Safety of Software with the TOUNDRA project on toolboxes for program proofs in cooperation with Calligramme. We also obtained a financial support from the region for the development of *CoLoR*.

8.2.1. Toundra

Participants: Isabelle Gnaedig, Hélène Kirchner.

The main goal of the Toundra action "Toolboxes for Program Proofs" is to develop integrated toolboxes for verifying and proving program properties of rule-based languages. Our aim is to provide expertise-encapsulated environments allowing non specialists to certify programs.

To achieve our goal, we first have to develop property proof algorithms for rewriting, that are specific to the context of programming. In fact, there already exists a large amount of proof techniques for rewriting properties but, most of the time, they are not adapted to the context of programming. Finer tools could allow

working with the initial semantics instead of the free one, and with specific strategies instead of the general rewriting relation.

Second, we have to study how these tools can cooperate, in the most efficient way, to cover the largest possible applicability domains and allow non-expert users.

To this end, we develop termination proof algorithms for specific strategies, like innermost and outermost strategies, local strategies on operators, or *ELAN*-like strategies. We also study applicability and complexity arguments on these tools, as well as their possible relations in order to develop an expertise kernel for managing the different proof tools.

8.3. National initiatives

We participate in the following GDR ALP projects: COLLAB (collaboration of solvers) and "Logic, Algebra and Computation" (mixing algebraic and logical systems).

8.3.1. Modulogic

Participants: Frédéric Blanqui, Horatiu Cirstea, Isabelle Gnaedig, Pierre-Etienne Moreau, Laika Moussa, Antoine Reilles.

Modulogic is an ACISI project started on September, 1st. Its main goal is to build an integrated toolbox for asserted software. This toolbox allows writing modules built on declaration, definitions, statements and proofs. Declarations can be refined into definitions and statements into proofs, by progressively migrating from the specification to the implementation, with inheriting and redefining mechanisms, and by instantiating parameters.

The INRIA Protheo team and the Mirho action, as well as the FOC group (SPI-LIP6, Paris 6, CEDRIC, CNAM) and the Cristal project are also involved.

The integrated toolbox we aim to build, will offer an appropriate interface for interactions with compilers of programming languages, with proof verifying systems, and with proof search systems, allowing to automatically refine statements. Definitions and some parts of the proofs will be let to the user. This toolbox will then be used for interacting with existing tools and languages (like Caml, *Coq* ...). The originality of our approach lies on a formal integration of these tools.

Contributions we would like to develop are the following: to conceive and realize the toolbox, to develop the component "proof research tool". They will be developed in order to be applied to safety strategies. In fact, we think that formal certification of safety properties can only be thought in a global way, and such a toolbox should help us to do it.

Building the toolbox can be seen as the continuation of the development of the systems Foc and *ELAN*. The ongoing work on the semantics of the object oriented languages and on the ρ-calculus will constitute the basis of the semantics for Modulogic. Building efficient proof research tools will be based on our work on the deduction modulo and on our expertise in the domain of rewriting. Adapting the toolbox to the safety strategies will be made possible thanks to our expertise for specifying safety properties in *Coq*, in particular the model of Bell and Lapadula.

8.4. International networks and working groups

We are involved in the COMPULOG network which consists of a lot of groups working on logic programming. We participate to the working groups *ERCIM Constraints* coordinated by F. Fages (INRIA project Contraintes, Rocquencourt) and *Programming Languages Technology* coordinated by N. Jones (Diku, Copenhague).

Olivier Bournez, Emmanuel Hainry and Paulin de Naurois are involved in the "Computability in Europe" network. This network aims at federating research efforts in Europe of teams working on computability and complexity, and in particular on new paradigms of computation such as analog computations.

We are involved in the thematic network APPSEM II on "Applied Semantics", which is coordinated by Martin Wirsing (LMU, Muenchen). This network is funded by the "5th Framework Programme" (FP5). We

contribute to the following themes of the network: proofs assistants, functional programming and dependent types; types and type inference in programming; resource models and web data (e.g. resource-bounded computation reasoning about linked data); continuous phenomena in Computer Science (e.g. computing with real numbers).

We participate to REWERSE - “Reasoning on the Web”, a Network of Excellence (NoE) within the “6th Framework Programme” (FP6), Information Society Technologies (IST). The main objective of this project is the development of a coherent and complete, yet minimal, collection of inter-operable reasoning languages for advanced Web systems and applications. These languages will be tested on context-adaptive Web systems and Web-based decision support systems selected as test-beds for proof-of-concept purposes. Finally, we aim at bringing the proposed languages to the level of open pre-standards amenable to submissions to standardization bodies such as the W3C.

8.5. International bilateral initiatives

Chili. Since 2002, we have a French-Chilean cooperation with the Federico Santa Maria Technical University of Valparaiso. This project, called COCARS and supported by CONICYT and INRIA, is about the use of rules and strategies for the design of constraint solvers.

Udine and **Venizia.** Clara Bertolissi carries out her PhD thesis in co-supervision with the Universities of Udine and Venizia. The supervisors are Furio Honsell at Udine and Paolo Baldan at Venizia (Italy) and Claude Kirchner, and Horatiu Cirstea at Nancy.

Lisbon. Olivier Bournez and Emmanuel Hainry cooperate with Manuel Campagnolo and Daniel Graça from Lisbon, under the framework of a PAI Pessoa Grant.

London. Clara Bertolissi, Horatiu Cirstea, Germain Faure and Claude Kirchner cooperate with Maribel Fernandez and Ian Mackie from King’s College (London) and with Gilles Dowek, Jean-Pierre Jouannaud and François-Régis Sinot from LIX-École Polytechnique (Paris), under the framework of a PAI Alliance Grant.

8.6. Invited lecturers

The program of the seminars is available at http://protheo.loria.fr/seminaires_en.html.

- Manuel Campagnolo (University of Lisbon), “Real recursive functions and their hierarchies: structural and computational complexity”
- Daniel Graça (University of Algarve, Portugal), “A digression over the General Purpose Analog Computer”
- Eduardo Bonelli (Stevens Institute of Technology, USA), “Information Flow Analysis in Typed Assembly Languages”
- François Lamarche (Calligramme, LORIA), “Lambda-termes, réseaux et sémantique des jeux”
- Yann Radenac (IRISA), “Un modèle de calcul chimique d’ordre supérieur étendu aux multi-ensembles infinis et hybrides”
- Luigi Liquori (INRIA Sophia Antipolis), “Intersection Types *à la* Church”
- Mariangiola Dezani (Università di Torino), “Boxed Ambients with Communication Interfaces”
- Herman Geuvers (Radboud Universiteit), “Natural Deduction via Graphs”
- Delia Kesner (PPS Jussieu), “Pure Pattern Calculus”
- Yann Régis-Gianas (INRIA Rocqencourt), “Inférence stratifiée pour les types algébriques gardés”

9. Dissemination

9.1. Leadership within scientific community

AFIT French chapter of EATCS

ASIAN Asian Computing Science Conference

CSL Conference of the European Association for Computer Science Logic

DCM Workshop on Developments in Computational Models

GTTSE Summer School on Generative and Transformational Techniques in Software Engineering

IEHSC International Embedded and Hybrid Systems Conference

IFIP International Federation for Information Processing

IFIP WG IFIP Working Group

JFLA French-speaking workshop on Applicative Languages

LDTA Language Descriptions, Tools and Applications

LICS International Conference on Logics in Computer Science

LPAR International Conference on Logic for Programming Artificial Intelligence and Reasoning

PPDP International Conference on Principles and Practice of Declarative Programming

QPQ Online journal for peer-reviewed source code for deductive software components

RTA International Conference on Rewriting Techniques and Applications

RULE International Workshop on Rule-Based Programming

SPECIF French society of professors and researchers in computer science

STACS Symposium on Theoretical Aspects of Computer Science

- Frédéric Blanqui:
  - Organization of the 3rd Nancy-Saarbrücken Workshop on Logic, Proofs and Programs, Nancy, 13-14 October (39 participants).
  - Organization of a QSL workshop on functional programming and verification, Nancy, 12 May 2005 (about 30 participants).
- Olivier Bournez:
  - Leader of the French node of the “Computability in Europe” network.
  - Leader of an ARASSIA on applications of game theory to security of algorithms for networks.
  - Co-organization with Manuel Campagnolo in Lisbon of a workshop on “Computations on the Continuum” on June 27-28th 2005.
  - Co-organization with Paulin de Naurois of a QSL workshop on other models of computations on March 24th 2005.
  - Member of the scientific committee of IEHSC’05.
- Horatiu Cirstea

  - Co-chair with Narciso Martí-Oliet (Universidad Complutense de Madrid) of the program committee of RULE'05.
  - Co-chair with Maribel Fernandez (King's College, Londres) of the program committee of the 2nd workshop on the rewriting calculus.

- Isabelle Gnaedig:

  - Member of the QSL coordination board.
  - Coordinator of the Protheo part of the ACISI project Modulogic.
  - Manager of the PRST-IL project Toundra.

- Claude Kirchner:

  - Chair of the scientific committee for the national ACISI programs
  - Chair of the evaluation committee of the ARASSIA.
  - Chair of the LORIA building extension committee.
  - Editorial boards of *Journal of Automated Reasoning*, *Journal of Information Science and Engineering*, *Journal of Applied Logic*.
  - Invited editor of the Information Security section of Vuibert's encyclopedia of Informatics and Information Systems [10], [21].
  - Program committees of STACS'05, RTA'05 and DCM'05.
  - Chair of the IFIP WG 1.6 on rewriting and applications.
  - Member of the advisory boards of LICS and PPDP.
  - Co-organizer of the "Symposium Franco-Japonais sur la sécurité informatique " (Tokyo, September 5–7)
  - Member of working groups of the CSD and the CISSI.

- Hélène Kirchner:

  - Director of LORIA and INRIA Lorraine.
  - Coordinator of the PRST-IL.
  - Editorial boards of *Annals of Mathematics and Artificial Intelligence*, *Computing and Informatics* and *Logical Methods in Computer Science*.
  - Editorial board of the QPQ forum on rewriting.
  - Program committees of LPAR'05.
  - Member of Scientific directorate of the Dagstuhl international conference center.

- Pierre-Etienne Moreau:

  - LORIA committee for computer and software equipments.
  - Program committee of GTTSE'05.
  - Co-chair with Thérèse Hardin of JFLA'05.
  - Steering committee of LDTA.

9.2. Teaching

We do not mention the teaching activities of the various teaching assistants and lecturers of the project who work in various universities of the region.

- Olivier Bournez:
  - M2 course on algorithmic verification at UHP.
  - M2 course on semantic of parallel and distributed systems at UHP
  - M1 course on algorithmic and computational complexity at UHP.
- Isabelle Gnaedig:
  - Coordinator of the courses on program and software specification at ESIAL and UHP (for DESS diploma).
  - Courses on algebraic specifications, the LOTOS language and the semantic of concurrent processes at ESIAL and UHP.
  - Course and supervised practical works on formal methods for specifying and validating software to 2nd year students of the engineering school “École des Mines”.
- Claude Kirchner:
  - Master course in Nancy on logic and automated theorem proving with Adam Cichon.
  - Course on deduction modulo at the Paris Master of Research in computer science (MPRI) (3h)
  - Master course in Nancy on programming and proving with rule based languages, with Pierre-Etienne Moreau.
- Pierre-Etienne Moreau:
  - Master course in Nancy on programming and proving with rule based languages, with Claude Kirchner.
  - Lectures at ESIAL on fundamental data-structures.
  - Java courses at IUT-Nancy2

9.3. Invited talks

- Horatiu Cirstea:
  - "Overview of the rewriting calculus", King's College, London, November.
- Claude Kirchner:
  - "On security research at INRIA", INRIA scientific committee, Sophia-Antipolis, March.
  - "Certification of pattern matching compiled code", 2nd Taiwanese-French Conference in Information Technologies, Tainan, Taiwan, March.
  - "Narrowing Based Inductive Proof Search", Workshop on Programming Logics in memory of Harald Ganzinger, Saarbrücken, June.
  - "An Overview of the rewriting calculus", IFIP, TC1 meeting, Chicago, June.
  - "Beyong deduction modulo", ICALP Workshop on Structures and Deduction, Lisbon, July.
  - "The rewriting calculus", Grenoble, December.
- Hélène Kirchner:
  - "Non-intrisive formal methods", AICA'05, Italy, October.

9.4. Thesis and admission committees

- Olivier Bournez:
  - AFIT PhD thesis award committee.
  - UHP recruitment committee (section 27), tenured since September.
  - INPL recruitment commitee (section 27), substitute since September.
  - Metz University recruitment committee (section 27), substitute since September.
- Horatiu Cirstea:
  - Clara Bertolissi "The graph rewriting calculus: properties and expressive capabilities", PhD
- Frédéric Blanqui:
  - Secretary of the SPECIF PhD thesis award committee.
- Isabelle Gnaedig:
  - ESIAL admission committee.
- Claude Kirchner:
  - Recruitment committee for CR INRIA in Lorraine.
  - Member of the post-doc selection committee of the STIC department of CNRS.

  - Claude Marché “Preuves mécanisées de propriétés de programmes”, HDR.
  - Benjamin Wack “Typage et déduction dans le calcul de réécriture”, PhD.
  - Clara Bertolissi “The graph rewriting calculus: properties and expressive capabilities”, PhD.

- Hélène Kirchner:

  - Recruitment committees (section 27) of UHP Nancy1, Nancy2, INPL.
  - Sébastien Limet “Représentation des langages de n-uplets d’arbres par des programmes logiques et applications”, HDR

- Pierre-Etienne Moreau:

  - Member of the UHP recruitment committee (section 27).
  - Participation to an INRIA recruitment committee for research engineers.

10. Bibliography

Major publications by the team in recent years

[1] G. BARTHE, H. CIRSTEA, C. KIRCHNER, L. LIQUORI. *Pure Patterns Type Systems*, in " Principles of Programming Languages - POPL2003, New Orleans, USA", ACM, Jan 2003, p. 250–261.

[2] F. BLANQUI. *Definitions by Rewriting in the Calculus of Constructions*, in "Mathematical Structures in Computer Science", vol. 15, n^o 1, Feb 2005, p. 37-92.

[3] O. BOURNEZ, E. HAINRY. *An analog Characterization of Elementarily Computable Functions Over the Real Numbers*, in "31st International Colloqiuim on Automata, Languages and Programming - ICALP’2004, Turku, Finland", J. DIAZ, J. KARHUMÄKI, A. LEPISTO, D. T. SANNELLA (editors). , Lecture Notes in Computer Science, vol. 3142, Springer, Jul 2004, p. 269-280.

[4] O. BOURNEZ, E. HAINRY. *Real Recursive Functions and Real Extensions of Recursive Functions*, in "Machines and Universal Computations - MCU’2004, Saint-Petersburg, Russia", Sep 2004.

[5] H. CIRSTEA, C. KIRCHNER. *The rewriting calculus - Part I and II*, in "Logic Journal of the Interest Group in Pure and Applied Logics", vol. 9, n^o 3, May 2001, p. 427-498.

[6] G. DOWEK, T. HARDIN, C. KIRCHNER. *Theorem Proving Modulo*, in "Journal of Automated Reasoning", vol. 31, n^o 1, Nov 2003, p. 33-72.

[7] O. FISSORE, I. GNAEDIG, H. KIRCHNER. *Outermost ground termination*, in "Proceedings of the Fourth International Workshop on Rewriting Logic and Its Applications, Pisa, Italy", Electronic Notes in Theoretical Computer Science, vol. 71, Elsevier Science Publishers B. V. (North-Holland), September 2002.

[8] H. KIRCHNER, P.-E. MOREAU. *Promoting Rewriting to a Programming Language : A Compiler for Non-Deterministic Rewrite Programs in Associative-Commutative Theories*, in "Journal of Functional Programming", vol. 11, n^o 3, Mar 2001, p. 207-251.

[9] P.-E. MOREAU, C. RINGEISSEN, M. VITTEK. *A Pattern Matching Compiler for Multiple Target Languages*, in "12th Conference on Compiler Construction, Warsaw (Poland)", G. HEDIN (editor). , LNCS, vol. 2622, Springer-Verlag, May 2003, p. 61–76.

Books and Monographs

[10] C. KIRCHNER (editor). *Sécurité informatique, Encyclopédie des systèmes d'information*, to appear, Vuibert, 2006.

Doctoral dissertations and Habilitation theses

[11] C. BERTOLISSI. *The graph rewriting calculus : properties and expressive capabilities*, Thèse de Doctorat, Institut National Polytechnique de Lorraine - INPL, Oct 2005.

[12] B. WACK. *Typage et déduction dans le calcul de réécriture*, Thèse de Doctorat, Université Henri Poincaré - Nancy I, Oct 2005.

Articles in refereed journals and book chapters

[13] F. BLANQUI. *Definitions by Rewriting in the Calculus of Constructions*, in "Mathematical Structures in Computer Science", vol. 15, n^o 1, Feb 2005, p. 37–92.

[14] F. BLANQUI. *Inductive types in the Calculus of Algebraic Constructions*, in "Fundamenta Informaticae", vol. 65, n^o 1-2, Mar 2005, p. 61–86.

[15] O. BOURNEZ. *How much can analog and hybrid systems be proved (super-)Turing*, in "Applied Mathematics and Computation", Oct 2005.

[16] O. BOURNEZ, F. CUCKER, P. JACOBÉ DE NAUROIS, J.-Y. MARION. *Implicit Complexity over an Arbitrary Structure : Quantifier Alternations*, in "Information and Computation", Oct 2005.

[17] O. BOURNEZ, F. CUCKER, P. JACOBÉ DE NAUROIS, J.-Y. MARION. *Implicit Complexity Over an Arbitrary Structure : Sequential and Parallel Polynomial Time*, in "Journal of Logic and Computation", vol. 15, n^o 1, Feb 2005, p. 41–58.

[18] O. BOURNEZ, E. HAINRY. *Elementarily Computable Functions Over the Real Numbers and R-Sub-Recursive Functions*, in "Theoretical Computer Science", Oct 2005.

[19] O. BOURNEZ, E. HAINRY. *Recursive Analysis Characterized as a Class of Real Recursive Functions*, in "Fundamenta Informaticae", Oct 2005.

[20] G. FAURE, C. KIRCHNER, H. CIRSTEA. *A Rho-Calculus of explicit constraint application*, in "Higher-Order and Symbolic Computation", Aug 2005.

[21] C. KIRCHNER. *Sécurité informatique, Encyclopédie des systèmes d'information, introduction*, to appear, Vuibert, 2006.

[22] M. VAN DEN BRAND, P.-E. MOREAU, J. VINJU. *A generator of efficient strongly typed abstract syntax trees in Java*, in "IEE Proceedings - Software Engineering", vol. 152, n° 2, Apr 2005, p. 70–78.

Publications in Conferences and Workshops

[23] C. BERTOLISSI. *The graph rewriting calculus : confluence and expressiveness*, in "9th Italian conference on Italian Conference on Theoretical Computer Science - ICTCS 2005, Siena, Italy", G. M. P. MARIO COPPO (editor). , Lecture Notes in Computer Science, vol. 3701, Springer Verlag, Oct 2005, p. 113–127.

[24] F. BLANQUI. *Decidability of Type-checking in the Calculus of Algebraic Constructions with Size Annotations*, in "14th Annual Conference of the EACSL, Oxford, UK", L. ONG (editor). , Lecture Notes in Computer Science, vol. 3634, Springer, Aug 2005, p. 135–150.

[25] O. BOURNEZ, F. CUCKER, P. JACOBÉ DE NAUROIS, J.-Y. MARION. *Logical Characterizations of $P_{\mathcal{K}}$ and $NP_{\mathcal{K}}$ Over an Arbitrary Structure K*, in "3rd APPSEM II Workshop - APPSEM'05, Frauenchiemsee, Germany", Sep 2005.

[26] O. BOURNEZ, F. GARNIER. *Proving Positive Almost-Sure Termination*, in "16th International Conference on Rewriting Techniques and Applications - RTA'2005, Nara, Japan", J. GIESL (editor). , Lecture Notes in Computer Science, vol. 3467, Springer Verlag, Apr 2005, p. 323–337.

[27] O. BOURNEZ, L. IBANESCU, H. KIRCHNER. *From Chemical Rules to Term Rewriting*, in "6th International Workshop on Rule-Based - RULE'05, Nara, Japan", to appear, Apr 2005.

[28] G. BUREL, C. KIRCHNER. *Completion is an Instance of Abstract Canonical System Inference*, in "23rd International Symposium on Theoretical Aspects of Computer Science - STACS 2006, Marseille, France", Soumis, Nicolas Ollinger, Feb 2005.

[29] H. CIRSTEA, G. FAURE, M. FERNANDEZ, I. MACKIE, F.-R. SINOT. *Rho-calculus for functional languages*, in "17th International Workshop on Implementation and Application of Functional Languages - IFL'05, Dublin, Ireland", Soumis, Sep 2005.

[30] H. CIRSTEA, C. KIRCHNER, L. LIQUORI, B. WACK. *Decidable Type Inference for the Polymorphic Rewriting Calculus*, in "17è Journées Francophones des Langages Applicatifs - JFLA 2006, Pauillac, France", Jan 2005.

[31] I. GNAEDIG, H. KIRCHNER. *Termination of rewriting strategies : a generic approach*, in "APPSEM II Workshop 2005, Chiemsee, Germany", M. HOFMANN, H.-W. LOIDL (editors). , APPSEM II & Ludwig Maximilians Universität München, Sep 2005.

[32] O. KAHRAMANOGULLARI, P.-E. MOREAU, A. REILLES. *Implementing Deep Inference in TOM*, in "ICALP Workshop on Structures and Deduction, Lisbon, Portugal", F. L. PAOLA BRUSCOLI, C. STEWART (editors). , Jul 2005, p. 158-172.

[33] C. KIRCHNER, H. KIRCHNER, A. SANTANA DE OLIVEIRA. *Anchoring modularity in HTML*, in "1st International Workshop on Automated Specification and Verification of Web Sites - WWV'05, Valencia, Spain", Mar 2005, p. 139–151.

[34] C. KIRCHNER, P.-É. MOREAU, A. REILLES. *Formal Validation of Pattern Matching Code*, in "Proceedings of the 7th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming - PPDP'2005, Lisbon, Portugal", P. BARAHONA, A. FELTY (editors). , ACM, Jul 2005, p. 187–197.

[35] H. KIRCHNER, S. RANISE, C. RINGEISSEN, D.-K. TRAN. *On Superposition-Based Satisfiability Procedures and their Combination*, in "2nd International Colloquium on Theoretical Aspects of Computing - ICTAC'05, Hanoi, Vietnam", D. V. HUNG, M. WIRSING (editors). , Lecture Notes in Computer Science, vol. 3722, Springer Verlag, Oct 2005, p. 594–608.

Internal Reports

[36] P. BALDAN, C. BERTOLISSI, H. CIRSTEA, C. KIRCHNER. *A rewriting calculus for cyclic higher-order term graphs*, Technical report, Nov 2005.

[37] E. BALLAND. *Optimisation du filtrage par transformations de programmes*, Stage de DEA, Jun 2005.

[38] F. BLANQUI, C. RIBA, C. KIRCHNER. *On the confluence of lambda-calculus with conditional rewriting*, Internal Report, Oct 2005.

[39] G. BUREL. *Systèmes Canoniques Abstraits : Application à la Déduction Naturelle et à la Complétion*, Rapport de stage, Aug 2005.

[40] H. CIRSTEA, E. COQUERY, W. DRABENT, F. FAGES, C. KIRCHNER, L. LIQUORI, B. WACK, A. WILK. *Types for REWERSE reasoning and query languages*, REWERSE Network of Excellence, Rapport de Contrat, 2005.

[41] I. GNAEDIG, H. KIRCHNER. *Termination of rewriting strategies : a generic approach*, Rapport de recherche, Jul 2005.

[42] H. REJEB. *Preuve inductive de terminaison faible : vers une implantation*, Rapport de stage, Jun 2005.

Miscellaneous

[43] C. KIRCHNER, H. KIRCHNER, F. NAHON. *Narrowing Based Inductive Proof Search*, June 2005, Presented at the Workshop on Programming Logics in memory of Harald Ganzinger.

Bibliography in notes

[44] E. ASARIN, O. MALER. *Achilles and the Tortoise Climbing Up the Arithmetical Hierarchy*, in "Journal of Computer and System Sciences", vol. 57, n^o 3, Dec 1998, p. 389–398.

[45] C. BERTOLISSI, P. BALDAN, H. CIRSTEA, C. KIRCHNER. *A rewriting calculus for cyclic higher-order term graphs*, in "2nd International Workshop on Term Graph Rewriting - TERMGRAPH'2004, Rome, Italy", M. FERNANDEZ (editor). , Electronic Notes in Theoretical Computer Science, Oct 2004.

[46] F. BLANQUI. *A type-based termination criterion for dependently-typed higher-order rewrite systems*, in "15th International Conference on Rewriting Techniques and Applications - RTA'04, Aachen, Germany", Jun 2004.

[47] O. BOURNEZ. *Complexité Algorithmique des Systèmes Dynamiques Continus et Hybrides*, Ph. D. Thesis, Ecole Normale Supérieure de Lyon, Jan 1999.

[48] O. BOURNEZ, E. HAINRY. *An analog Characterization of Elementarily Computable Functions Over the Real Numbers*, in "2nd APPSEM II Workshop - APPSEM'2004, Tallinn, Estonia", Apr 2004, http://www.loria.fr/publications/2004/A04-R-289/A04-R-289.ps.

[49] O. BOURNEZ, E. HAINRY. *Real Recursive Functions and Real Extensions of Recursive Functions*, in "Machines and Universal Computations - MCU'2004, Saint-Petersburg, Russia", Sep 2004.

[50] O. BOURNEZ, C. KIRCHNER. *Probabilistic rewrite strategies. Applications to ELAN*, in "13th International Conference on Rewriting Techniques and Applications - RTA'2002, Copenhagen, Denmark", S. TISON (editor). , Lecture Notes in Computer Science, vol. 2378, Springer, Jul 2002, p. 252-266.

[51] H. CIRSTEA, E. COQUERY, W. DRABENT, F. FAGES, C. KIRCHNER, J. MALUSZYNSKI, B. WACK. *Types for Web Rule Languages : a preliminary study*, Rapport Intermédiaire, Sep 2004, http://www.loria.fr/publications/2004/A04-R-560/A04-R-560.ps.

[52] H. CIRSTEA, G. FAURE, C. KIRCHNER. *A rho-calculus of explicit constraint application*, in "Workshop on Rewriting Logic and Applications, Barcelona (Spain)", Electronic Notes in Theoretical Computer Science, Mar 2004.

[53] H. CIRSTEA, C. KIRCHNER, M. MOOSSEN, P.-E. MOREAU. *Production Systems and Rete Algorithm Formalisation*, Rapport Intermédiaire, Oct 2004, http://www.loria.fr/publications/2004/A04-R-546/A04-R-546.ps.

[54] H. CIRSTEA, C. KIRCHNER, M. MOOSSEN, P.-E. MOREAU. *Production Systems and Rewrite Systems*, Rapport Intermédiaire, Nov 2004, http://www.loria.fr/publications/2004/A04-R-563/A04-R-563.ps.

[55] H. CIRSTEA, L. LIQUORI, B. WACK. *Rewriting Calculus with Fixpoints: Untyped and First-order Systems*, in "Types for Proofs and Programs (TYPES), Torino (Italy)", S. BERARDI, M. COPPO, F. DAMIAN (editors). , Lecture Notes in Computer Science, vol. 3085, May 2003, p. 147–171.

[56] G. COLATA. *With Major Math Proof, Brute Computers Show Flash of Reasoning Power*, in "The New York Times", Tuesday December 10, 1996.

[57] COQ-DEVELOPMENT-TEAM. *The Coq Proof Assistant Reference Manual - Version 8.0*, 2004, http://coq.inria.fr/, INRIA Rocquencourt, France.

[58] S. COUPET-GRIMAL, W. DELOBEL. *Une Preuve Effective de la Bonne Fondaison de l'Ordre R cursif Multi-ensemble sur les Chemins*, in "17e Journées Francophones des Langages Applicatifs", 2006.

[59] E. DEPLAGNE, C. KIRCHNER. *Induction as Deduction Modulo*, Rapport de recherche, Nov 2004, http://www.loria.fr/publications/2004/A04-R-468/A04-R-468.ps.

[60] N. DERSHOWITZ, C. KIRCHNER. *Abstract Saturation-based Inference*, in "Eighteenth Annual

IEEE Symposium on Logic in Computer Science - LICS'2003), Ottawa, Canada", Jun 2003, http://www.loria.fr/publications/2003/A03-R-422/A03-R-422.ps.

[61] D. DOLIGEZ. *Zenon: an automatic theorem prover for first-order logic*, 2004, http://focal.inria.fr/download.

[62] D. DOUGHERTY. *Adding Algebraic Rewriting to the Untyped Lambda Calculus*, in "Information and Computation", vol. 101, n° 2, 1992, p. 251-267.

[63] G. ETESI, I. NÉMETI. *Non-Turing computations via Malament-Hogarth space-times*, in "International Journal Theoretical Physics", vol. 41, 2002, p. 341–370.

[64] O. FISSORE, I. GNAEDIG, H. KIRCHNER. *CARIBOO : An induction based proof tool for termination with strategies*, in "Proceedings of the Fourth International Conference on Principles and Practice of Declarative Programming, Pittsburgh (USA)", ACM Press, Oct 2002, p. 62–73.

[65] O. FISSORE, I. GNAEDIG, H. KIRCHNER. *A proof of weak termination providing the right way to terminate*, in "First International Colloquium on Theoretical Aspects of Computing, Guiyang, Chine", Lecture notes in Computer Science, Springer Verlag, Sep 2004.

[66] S. HINDERER. *Certification des preuves de terminaison par interprétations polynomiales*, Stage de DEA, Loria, Jun 2004, http://www.loria.fr/publications/2004/A04-R-489/A04-R-489.ps.

[67] M. L. HOGARTH. *Does General Relativity Allow an Observer to View an Eternity in a Finite Time?*, in "Foundations of Physics Letters", vol. 5, 1992, p. 173–181.

[68] J.-P. JOUANNAUD, C. KIRCHNER. *Solving equations in abstract algebras: a rule-based survey of unification*, in "Computational Logic. Essays in honor of Alan Robinson, Cambridge (MA, USA)", J.-L. LASSEZ, G. PLOTKIN (editors). , chap. 8, The MIT press, 1991, p. 257-321.

[69] D. E. KNUTH, P. B. BENDIX. *Simple word problems in universal algebras*, in "Computational Problems in Abstract Algebra, Oxford", J. LEECH (editor). , Pergamon Press, 1970, p. 263-297.

[70] A. KOPROWSKI. *Well-foundedness of the Higher-Order Recursive Path Ordering in Coq*, Technical report, Free University of Amsterdam, The Netherlands, and Warsaw University, Poland, 2004.

[71] L. LIQUORI, B. WACK. *The Polymorphic Rewriting Calculus : Type checking vs. Type inference*, in "5th International Workshop on Rewriting Logic and its Applications - WRLA 2004, Barcelona, Spain", N. MARTI-OLIET, M. CLAVEL, A. VERDEJO (editors). , Electronic Notes in Theoretical Computer Science, Elsevier, Narciso Marti-Oliet, Mar 2004, http://www.loria.fr/publications/2004/A04-R-386/A04-R-386.ps.

[72] W. MCCUNE. *Solution of the Robbins Problem*, in "JAR", vol. 19, n° 3, 1997, p. 263-276.

[73] J. MESEGUER. *Conditional rewriting logic as a unified model of concurrency*, in "TCS", vol. 96, n° 1, 1992, p. 73-155.

[74] C. MOORE. *Recursion Theory on the Reals and Continuous-time Computation*, in "Theoretical Computer Science", vol. 162, 1996, p. 23-44.

[75] C. MOORE. *Recursion theory on the reals and continuous-time computation*, in "Theoretical Computer Science", vol. 162, nº 1, Aug 1996, p. 23–44.

[76] P.-E. MOREAU, C. RINGEISSEN, M. VITTEK. *A Pattern Matching Compiler for Multiple Target Languages*, in "International Conference on Compiler Construction - CC'2003, Varsovie, Pologne", G. HEDIN (editor). , Lecture notes in Computer Science, vol. 2622, Apr 2003, p. 61-76.

[77] F. MÜLLER. *Confluence of the lambda calculus with left-linear algebraic rewriting*, in "Information Processing Letters", vol. 41, nº 6, 1992, p. 293-299.

[78] TERESE. *Term Rewriting Systems*, M. Bezem, J. W. Klop and R. de Vrijer, eds., Cambridge University Press, 2002.

[79] B. WACK. *The Simply-typed Pure Pattern Type System Ensures Strong Normalization*, in "3rd IFIP International Conference on Theoretical Computer Science - TCS'2004, Toulouse, France", J.-J. LÉVY (editor). , Kluwer Academic Publishers, Aug 2004, p. 633-646.