INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team PROTHEO

Constraints, Mechanized Deduction and Proofs of Software Properties

Lorraine

THEME 2A

Activity Report

2003

Table of contents

1. Team

***PROTHEO** is a join 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).*

Glossary

DR	Director of research
CR	Research scientist
MC	Associate professor
ATER	Full-time teaching assistant
Moniteur	Half-time teaching assistant
MENRT	Grant from the french government
ENS	“Ecole Normale Supérieure”
ESIAL	Engineering School in Computer Science and Applications of Lorraine
DEA	Master diploma

Head of project-team
Claude Kirchner [DR INRIA]

Administrative assistants
Chantal Llorens [AI CNRS]

Research scientists
Frédéric Blanqui [CR INRIA, from 01/10]
Olivier Bournez [CR INRIA]
Isabelle Gnaedig-Antoine [CR INRIA]
Hélène Kirchner [DR CNRS, part time]
Pierre-Etienne Moreau [CR INRIA]
Christophe Ringeissen [CR INRIA]

Faculty members
Horatiu Cirstea [MC Nancy2]
Carlos Castro [MC UHP, part time]

Technical staff
Julien Guyon [INRIA fixed-term engineer]

Ph. D. students
Clara Bertolissi [MENRT]
Eric Deplagne [ATER, until 10/31]
Germain Faure [ENS Cachan, from 10/01]
Olivier Fissore [CNRS and Lorraine]
Florent Garnier [INRIA, from 11/01]
Emmanuel Hainry [ENS Lyon, from 10/01, with the INRIA project Calligramme]
Liliana Ibănescu [INPL]
Paulin Jacobé de Naurois [MENRT, with the INRIA project Calligramme]
Fabrice Nahon [High school teacher]
Quang-Huy Nguyen [ATER, until 08/31]
Antoine Reilles [CNRS, from 10/01]
Duc Tran [MENRT, from 10/01]

Benjamin Wack [MENRT]

Visiting scientists

Nachum Dershowitz [12/01-12/07]
Gopalan Nadathur [from 10/01]
Anamaria Martins Moreira [until 03/31]
Michael Mossens [09/15-09/20]
Marina Cristina Riff [07/01-07/22]
Anderson Santana [until 02/28]

Graduate and DEA student interns

Guillaume Burel [ENS Lyon, 06/13-07/21]
Issam Chebbi [DEA Nancy, 01/22-07/08]
Guillaume Darmont [ESIAL, 07/01-08/31]
Régis Durand [ESIAL, 06/23-08/15]
Germain Faure [DEA Paris, 04/01-09/31, with the INRIA project Lande]
Emmanuel Hainry [DEA Lyon, 02/15-07/08]
Pierre Henninger [ESIAL, 06/23-08/15]
Stéphane Hion [DEA Nancy, 01/22-07/08]
Stéphane Hrvatin [ESIAL, 07/01-08/31]
Vikram Krisna Prasad [IIT Kanpur, India, 05/12-07/19]
Damien Regnault [ENS Lyon, 06/02-07/13]
Antoine Reilles [DEA Nancy, 01/22-07/08]
Alexis Saettler [ESIAL, 07/01-08/31]
Duc Tran [DEA Nancy, 01/22-07/08]

2. Overall Objectives

The PROTHEO project aims at designing and implementing tools for program specification, proof of properties and safe and efficient execution.

We are working on an environment 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 themes: constraint solving, mechanized deduction with rewrite rules and strategies, and theorem proving.

- Constraint solving.
  - Constraint satisfiability on symbolic and numerical domains.
  - Collaboration of constraint solvers.
  - Constraint satisfaction techniques.
- Mechanized deduction with rewrite rules and strategies.
  - Strategy language for constraint solvers and theorem provers.
  - Non-deterministic computations.
  - Compilation of rewriting and strategies.
  - Properties of confluence, termination, modularity.
- Theorem proving.

– Deduction modulo.
– Proofs by induction.
– Observational proofs.
– Proofs of program properties.

The team develops several software packages detailed later in this document, that 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

Key words: *rewriting*, *strategies*, *rewrite based programming*, *functional programming*.

Rewriting is one of the main themes of the project being considered and studied in order to model computation (in case of confluent and terminating rewrite systems) or deduction in which case non specific requirements are needed. Rewrite rules could be used to define the notions of a relation [38], a logic [60] or a calculus [4]. Strategies guiding the application of rewrite rules have been studied since a long time and appear to be central in the definition of efficient evaluation tools as well as in the definition of deduction processes. The ρ-calculus gives us the general framework on which to ground our works on this topic.

The rewriting techniques have been developed since the '70s 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 [62].

Originally, the goal was to find a canonical rewriting system which allows one to prove the validity of an equational theorem by rewriting each member of the equality into the same term. Starting from an equational theory, the completion procedure of Knuth and Bendix [57] was conceived to generate, when that is possible, a confluent and terminating rewriting system. These two properties ensure the completeness of this method for deciding the validity of an equational theorem.

The 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 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 [4]. We have been studying the expressiveness of this general formalism and the properties of its various instances. We currently explore different type versions of the calculus and we analyze the possibility to generalize the deBruijn-Curry-Howard isomorphism.

3.2. Constraints

Key words: *constraint solving*, *satisfiability*, *combination problem*.

The study of satisfiability and solving of constraint systems on term or on specific domains is central in computation and in deduction. Combination techniques are of fundamental use to decompose a problem in simpler ones. The cooperation of constraint solvers or deductive systems could also be used to break down the difficulty of solving large problems.

The notion of constraint has shown its interest 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 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 [54]. 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

Key words: *deduction*, *rewriting*, *induction*, *constraints*, *paramodulation*, *resolution*.

The combination of constraint solving and rewriting techniques is present in many if not all systems of automated deduction. The study of their consistent combination and the design of complete and efficient theorem provers based on them leads to the possibility of checking program properties in a more systematic way. For instance, Noetherian rewrite systems allow to perform inductive reasoning based on their underlying terminating ordering. Constraint satisfaction techniques allow to delay the solving of constraint problems created by deduction steps.

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 lies on inductive reasoning or equation handling, and uses operator properties like associativity or commutativity.

Rewriting, which allows 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 developing specific termination proof tools for rewriting strategies.

Constraints allow to postpone complex symbolic problem solving, in order they can be solved in an expedient way. They also allow 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 to obtain much simpler automated proofs. Implementing these ideas has indeed allowed W. McCune [49][59] to solve an open mathematical problem. Combining constraints and rewriting based simplifications induces complex problems, as theoretical, for completeness of strategies, as practical, for efficiency of implementation. We explore these techniques both with a theoretical point of view and an experimental one.

4. Application Domains

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

Our researches apply 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 an ease in expressivity of 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 7.1).

5.2. ELAN

Key words: *rules*, *strategies*, *specification*, *computation*, *deduction*.

Participants: Horatiu Cirstea, Olivier Fissore, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau, Quang-Huy Nguyen, Christophe Ringeissen.

The **ELAN** [44] 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 takes from functional programming the concept of abstract data types and the function evaluation principle based on rewriting. But rewriting is inherently non-deterministic since several rules can be applied at different positions in a term and, in **ELAN**, a computation may have several results. This aspect is taken into account through choice operations and a backtracking capability. One of the main originality of the language is to provide strategy constructors to specify whether a function call returns several, at-least one or only one result. This declarative handling of non-determinism is part of a strategy language allowing the programmer to specify the control on rules application. This is in contrast to many existing rewriting-based languages where the term reduction strategy is hard-wired and not accessible to the designer of an application. The strategy language offers primitives for sequential composition, iteration, deterministic and non-deterministic choices of elementary strategies that are labeled rules. From these primitives, more complex strategies can be expressed. In addition the user can introduce new strategy operators and define them by rewrite rules. Evaluation of strategy application is itself based on rewriting. So the simple and well-known paradigm of rewriting provides both the logical framework in which deduction systems can be expressed and combined, and the evaluation mechanism of the language.

ELAN is documented, maintained and available at http://elan.loria.fr.

5.3. TOM

Key words: *pattern matching*, *rule based programming*, *compilation*.

Participants: Pierre-Etienne Moreau, Julien Guyon, Antoine Reilles, Christophe Ringeissen.

Since 2002, we have developed a new system called **TOM**, presented in [27]. This system consists in 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 [55]. 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.

6. New Results

6.1. Rewriting calculus

Key words: *rewriting*, *strategies*, *types*.

Starting from our previous works [43], we noticed that the tool necessary to control rewriting must be explicit and can be described quite naturally by using rewriting too. This observation brought us to a new calculus generalizing λ-calculus and rewriting which we called ρ-calculus [4].

We have proposed and studied different instances of this calculus and their corresponding type systems. We have shown the expressive power of this calculus by comparing it with higher-order rewriting systems and by giving a simple and typed encoding of standard strategies used in first-order rewriting. A version of the calculus handling explicitly matching and substitution applications has been also proposed.

6.1.1. The expressive power of the ρ-calculus

Participants: Clara Bertolissi, Horatiu Cirstea, Claude Kirchner.

We have already shown that λ-calculus and rewriting are generalized by the ρ-calculus, in the sense that the syntax and the inference rules of the ρ-calculus can be restricted to obtain the other two formalisms.

In the prolongation of our works on the expressive power of the ρ-calculus we have analyzed the relation between ρ-calculus and higher order rewriting. More precisely, we provided a direct translation of *CRS* into the ρ-calculus [15] and we showed that for every reduction in a *CRS* [56] there is a correspondent reduction in the ρ-calculus.

Starting from the classical untyped ρ-calculus we have proposed an extension with an enhanced expressive power that deals not only with terms but also with graphs and graph transformations.

The syntax of the ρ-calculus is adapted in order to describe graphs and the related notions such as sharing and cycles. The evaluation rules are also adapted for this new syntax leading to a calculus that we call *graph rewriting calculus*. In this new formalism we can represent and manipulate elaborated objects like, for example, regular infinite entities.

Graph transformations are performed by application of rewrite rules at the object level. Cycles and sharing make the matching algorithm more complicated. Starting from the definition of graph bisimulation, we propose a matching algorithm that exhibits a substitution, if it exists, as solution of a matching problem.

6.1.2. 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). Actual implementations have to take a much more realistic view: computations needed to find the solutions of a matching equation can be really important in some matching theories and the application of a substitution involves a structure traversal.

Following the works on explicit substitutions for λ-calculus, we propose, study and examplify in [34] a ρ-calculus with explicit constraint handling, including substitution applications. The approach is really modular, allowing extension to arbitrary matching theories. We prove that the calculus is powerful enough to deal with errors. We also prove the confluence of the calculus and the termination of the explicit constraint handling part. We tried also to give the intuition of how programs can be represented in the ρ-calculus taking examples of `ML` programs: we use the ρ-calculus as a possible theoretical back-end for an implementation.

We intend to extend the calculus to allow for the composition of substitutions. Moreover, we would like to take advantage of the very general management of errors in order to propose a named exception mechanism where errors could be labeled.

6.1.3. Typed programming with ρ-calculus

Participants: Horatiu Cirstea, Benjamin Wack, Claude Kirchner.

In collaboration with Luigi Liquori, we have extensively studied a first-order typed ρ-calculus *a la* Church called $\rho_{\rightarrow}^{Fix}$, whose type system is mainly inspired by simply typed λ-calculus. The type system enjoys subject reduction, type uniqueness and decidability of typing. Moreover, the type system of $\rho_{\rightarrow}^{Fix}$ allows one to type (object oriented flavored) fixpoints, leading to an expressive and safe calculus. In particular, using pattern matching, one can encode and type-check term rewriting systems in a natural and automatic way, as sketched in [11].

Early versions of the (untyped) ρ-calculus have already been used to describe the implicit (leftmost-innermost) and user defined strategies of **ELAN**[4] but some *ad hoc* operators were added to the basic calculus for this. The $\rho_{\rightarrow}^{Fix}$ presented here refers essentially to a simpler formulation of previously introduced versions of the rewriting calculus [48] where no new constructions have to be defined for expressing strategies for quite a large class of rewriting systems. The price to pay for this simplicity is that the encoded rewriting systems we handle are not the most general ones but still the most used ones in practice.

The ability to type-check term rewriting systems and strategies ensures a good trade-off between expressiveness and safety of programs: the type system of $\rho_{\rightarrow}^{Fix}$ is suitable for static analysis, *i.e.* it ensures that functions get arguments of the expected type. 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.

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 [14].

6.1.4. Typed ρ-calculi for logics

Participants: Horatiu Cirstea, Benjamin Wack, Claude Kirchner.

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 [14], 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 and confluence, and we conjecture strong normalization for well-typed terms.

The abstraction operators for terms and types are separate in P^2TS but, given the expressive power of the ρ-abstractor, we proposed a more elaborate type system where the abstraction on types is also described by rewriting [47]. By unifying the abstraction symbols for terms and types, we loose the unicity of types but we can characterize terms having an arbitrary but finitary number of types. Nevertheless, we show the unicity of types for some systems, such as the polymorphic ρ-calculus [48]. We also show that typing is preserved by reduction for all the proposed type systems.

We consider this work as a contribution in the study of correspondences between rewriting and logics. In this context, we can associate rewriting-based programs to ρ-terms and we are interested in finding a suitable logic as a Curry-Howard isomorphism between typed ρ-terms and this logic.

6.2. Rule based programming

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 of pattern-matching

Participants: Julien Guyon, Pierre-Etienne Moreau, Christophe Ringeissen.

In collaboration with Marian Vittek, we have designed a new formalism, called **TOM**, which allows us to integrate pattern matching facilities into imperative languages such as C, Java or Eiffel. This new formalism has been presented in [27].

TOM does not really define a new language: it is rather 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 Eiffel. The compilation process consists of translating new matching constructs into the underlying native language. An important feature of **TOM** is to support equational matching. In particular, list matching, also known as associative matching with neutral element. Recently, we have introduced an XML extension which allows us to manipulate XML documents and to retrieve information via associative matching.

When developing an application which manipulates tree-like data structures (such as symbolic computation, compilation, XML transformation, message exchange, etc.), the need for matching can rarely be avoided, simply because some information has to be retrieved. **TOM** is particularly well suited for describing and implementing various kinds of applications where extraction of information has to be performed.

6.2.2. Representation of abstract data-types

Participant: Pierre-Etienne Moreau.

In collaboration with Olivier Zendra from the DESIGN team, we have studied the garbage collector algorithm provided by the standard C implementation of the ATerm Library and stressed the fact that some peculiarities of this functional library could be taken advantage of by the memory management system. We have designed and implemented GC^2, a new mark-and-sweep generational garbage collector for the ATerm Library that builds upon these peculiarities. This implementation improves the efficiency of all programs that use the ATerm Library. The source code is integrated into the main CVS Tree and is used to build a large number of tools.

In collaboration with Mark van den Brand and Jurgen Vinju of CWI at Amsterdam, we have presented a Java back-end for ApiGen, 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 sub-term 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 sub-term sharing in Java. Practical experience shows that this approach can not only be used for tools that are otherwise manually constructed, but also for internal data-structures in generated tools.

6.2.3. Environments for rule based programming

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

In collaboration with Mark van den Brand and Jurgen Vinju, from CWI, we have developed an environment for quickly implementing the syntax and semantics of term rewriting based formalisms [30]. Based on the Meta-Environment (an open architecture of tools, libraries, user-interfaces and code generators targeted to the design and implementation of term rewriting environments). We show that by using this mature programming environment, a new term rewriting formalism can be obtained in a few steps. Our approach is based on well-known software engineering concepts: standardization (of architecture and exchange format), software reuse (component based development), source code generation and parameterization. This approach is the basis of the new **ELAN**-4 system.

In parallel to this approach, we have studied the design of a library which allows the definition of search strategies. As rewriting is inherently concurrent, rewriting rules have been used to express the implicit parallelism of functional programs. In rewriting logic, rewriting rules can also express parallel and non-deterministic applications. We present in [36] a system of strategies for controlling concurrent application of rewriting rules and giving support for parallel evaluations of rewriting systems. We have studied the equivalences between composed strategies and the possibilities of simplifications for these composed strategies. These simplifications lead to optimized strategies evaluations, transforming the initial strategy into a strategy whose evaluation maximizes the possibilities of parallel evaluations. We also have proposed a way of compiling **ELAN** strategies into these parallel strategies, using concurrency to deal with non-determinism. An implementation for these strategies operators has been added to the **TOM** runtime library.

6.2.4. Program analysis and certification of program transformations

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

During his visit to CWI, Antoine Reilles used the ASF+SDF Meta-Environment for analyzing programs written in Java+**TOM**, using the RSCRIPT formalism, which is a scripting language based on the relational calculus, developed at CWI. The formalism provides operators for creating and manipulating comprehensions, enumerations, composition of two relations, cartesian product or transitive closure. With RSCRIPT, it becomes possible to derive additional information from relations extracted from a source code. By extending an existing Java parser written in SDF, we have proposed a methodology to extract informations from both the Java and the **TOM** formalism. This approach could be applied to compute the type of an hybrid expression composed of Java and **TOM**.

In parallel, we are also studying the certification of the transformation of patterns from the **TOM** language into an imperative target language, like Java. 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.

6.2.5. XML and rewriting

Participants: Guillaume Darmont, Claude Kirchner, Pierre-Etienne Moreau, Jürgen Stuber.

In this sub-project, two different paths have been followed in parallel. In collaboration with Mark van den Brand, we have used SGLR parsing and term rewriting with **ELAN**-4 to extract the semantics of mathematical formulas from a LaTeX document and representing them in MathML. The LaTeX document we used is part of the Digital Library of Mathematical Functions (DLMF) project of the US National Institute of Standards and Technology (NIST) and obeys project-specific conventions, which contains macros for mathematical constructions, among them 200 predefined macros for special functions, the subject matter of the project. The SGLR parser can parse general context-free languages, which suffices to extract the structure of mathematical formulas from calculus that are written in the usual mathematical style, with most parentheses and multiplication signs omitted. The parse tree is then rewritten into a more concise and uniform internal syntax that is used as the base for extracting MathML or other semantical information. This has been presented in [29].

In order to improve the expressiveness of our tools, we have also studied the integration of XML pattern matching facilities into the **TOM** system. A technical report is in preparation.

6.2.6. Tool support for reusing ELAN rule-based components

Participants: Anamaria Martins Moreira, Christophe Ringeissen, Anderson Santana.

The adaptation of software components developed for a specific application in order to generate reusable components often includes some kind of generalization. This generalization may be carried out, for instance, by the renaming of some identifiers or by its parameterization. In our work, we are specially interested in the generalization by parameterization of algebraic specification components. Generalization and some other transformations on algebraic specifications are being integrated in the FERUS tool. This tool was initially developed for the Common Algebraic Specification Language (CASL). We show in [26] its adaptation to the new version of the rule-based programming language **ELAN**.

6.2.7. Representing communicating processes in ELAN

Participants: Horatiu Cirstea, Stéphane Hion, Claude Kirchner.

Designing large distributed and heterogeneous systems is a difficult task and the interoperability of the different software components seems to be the solution. Various solutions based on different concepts were developed like, for example, .NET, SOAP, CORBA and Toolbus [41].

One can notice that most of these models are not based on a logical formalism such as process algebras or the π-calculus. The Toolbus, for its part, is based on the algebra of communicating processes (ACP) [42], an algebraic approach for the description of parallel and communicating processes.

Moreover, the Toolbus was partly specified in the environment ASF+SDF [63] that allows one to describe the operational semantics of a language by rewriting rules and to prototype the language by the application of these rules. It thus appears interesting to study how rewriting, **ELAN** and its strategies can be used to model the algebra of communicating processes and the functionalities of the Toolbus.

We thus have developed a software bus based on rewriting [35]. Starting from the study of an existing software bus, we proposed a rewriting system which models the algebra of communicating processes. This system was shown terminating and confluent. We have defined different strategies guiding the application of the rules and we obtained a better control of the reduction steps and thus, an efficient implementation. This prototype can be used, on the one hand, as a verification tool (*e.g.* for the detection of *dead-lock* or for checking the process (traces) equivalence) and, on the other hand, as an executable bus.

6.3. Constraints

Key words: *matching*, *combination*.

We are studying the design of symbolic constraint solvers and decision procedures. Among the constraint solving problems we are interested in, we concentrate our efforts on some matching problems. We study first-order equational matching as well as a form of higher-order matching of interest to understand the matching process used by the XSLT language, which aims at transforming XML documents thanks to the application of rules. Also, we continue our study on the combination of constraint solvers, when the corresponding theories may have non-disjoint signatures. In particular, we focus on the combination problems for satisfiability procedures and matching algorithms.

6.3.1. Unions of non-disjoint theories and combinations of satisfiability procedures

Participant: Christophe Ringeissen.

In [13], we outline a theoretical framework for the combination of decision procedures for constraint satisfiability. We describe a general combination method which, given a procedure that decides constraint satisfiability with respect to a constraint theory T_1 and one that decides constraint satisfiability with respect to a constraint theory T_2, produces a procedure that (semi) decides constraint satisfiability with respect to the union of T_1 and T_2. We provide a number of model-theoretic conditions on the constraint language and the component constraint theories for the method to be sound and complete, with special emphasis on the case in which the signatures of the component theories are non-disjoint. We also describe some general classes of theories to which our combination results apply, and relate our approach to some of the existing combination methods in the field.

This research has been carried out in cooperation with Cesare Tinelli (University of Iowa).

6.3.2. Matching in a class of combined non-disjoint theories

Participant: Christophe Ringeissen.

In [28], we present new equational matching and unification results in some combinations of non-disjoint equational theories. Some results are already known for theories sharing an appropriate notion of constructors. We investigate the idea of considering theories that are not explicitly based on the notion of constructors. In this direction, a new class of theories is presented, where a theory is defined as a union of two sub-theories, one such that shared symbols do not affect the behavior of the theory, and another one given by a term rewrite system on shared symbols. Matching and unification problems are studied for this class of theories, and for unions of theories in this class. Results obtained for the matching problem are particularly relevant.

6.3.3. Context matching problems

Participant: Jürgen Stuber.

Initially motivated by the precise definition of the matching process in the XSLT language, and in collaboration with Manfred Schmidt-Schauß from J.-W. Goethe Universität in Frankfurt (Germany), we give algorithms for linear and for general context matching and discuss how the performance in the general case can be improved through the use of information derived from approximations that can be computed in polynomial time. We investigate the complexity of context matching with respect to the stratification of variable occurrences, where our main results are that stratified context matching is NP-complete, but that stratified simultaneous monadic context matching, SSMCM for short, is in P. SSMCM is equivalent to stratified simultaneous word matching. We also show that the linear and the shared-linear case are in P and of time complexity $O(n^3)$, and that varity 2 context matching, where variables occur at most twice, is NP-complete.

6.4. Rewriting and strategies

Key words: *rules*, *strategies*, *constraints*, *decision procedures*.

This section considers how to extend the notion of rewriting controlled by strategies, and how to apply rules and strategies to design constraint solvers and decision procedures. In particular, we continue the study of probabilistic strategies, and we develop a rule language to specify the interaction between constraint solvers, and we apply a rule-based proof methodology initially introduced for unification problems to design rule-based decision procedures for the satisfiability problem in union of some signature-disjoint theories. From our expertise in the rule-based design of constraint solvers and provers, we continue the setting of a general methodology for rule-based programming.

6.4.1. Rewriting and probabilities

Participants: Olivier Bournez, Guillaume Burel, Mathieu Hoyrup, Claude Kirchner.

Rewriting Logic [60] is known to provide a very elegant and powerful framework for unifying a wide variety of models, including concurrency models and deduction systems. Indeed, the basic axioms of this logic, which are rewrite rules can be viewed in two dual ways: *computationally*, they can be viewed as the local transition of a concurrent system or *logically*, they can be viewed as the inference rule of some logic.

In order to extend the modeling capabilities of rule based languages, it seems natural to extend the framework with probabilities: for example, the modeling of concurrent systems often requires to consider that local transitions can be subject to some probabilistic laws.

In a paper presented at RTA'2002, *strategies* were shown to provide a nice setting for expressing probabilistic choices in rule based languages. *Probabilistic abstract reduction systems* and notions like *almost-sure termination* or *probabilistic confluence* were introduced and related to the classical notions.

This year, we presented a paper [20] devoted to a next step: understand whether there exists a valid and useful notion of rewrite system and rewriting logic in presence of probabilities.

We proved that there is no hope to build a sound and complete proof system that would prove whether two terms are in relation by the reflexive transitive closure of the reduction relation of a given rewrite system with associated probabilities in the general case.

We proposed a notion of probabilistic rewriting logic. One important difference between the proposed setting and the classical rewriting logic setting is that proof terms become now mandatory, in order to have completeness results: we prove that when proof terms are present, probabilistic rewriting logic is sound and complete.

We also show that the proposed probabilistic rewriting logic extends the modeling capabilities of classical rewriting logic on simple examples.

Through Guillaume Burel's internship [31], we started to understand whether there exists a valid notion of equational proof theory in presence of probabilities. We tried to understand which results about first-order probabilistic logics à la Halpern are valid when restricting first-order theories to equational theories.

6.4.2. *Rules and strategies for generation of chemical reactions*

Participants: Hélène Kirchner, Olivier Bournez, Liliana Ibănescu.

Several software systems have been developed recently for the automated generation of combustion reaction kinetic mechanisms using different representations of species and reactions and different generation algorithms. In parallel, several software systems based on rewriting have been developed for the easy modeling and prototyping of systems using rules controlled by strategies. We used the rewrite environment **ELAN** for modeling the automated generation of combustion reaction mechanisms. A previous implementation was done by chemists in the **EXGAS** kinetic mechanism generator system but could not deal with cyclic molecules. With an adequate data type for representing molecules and reactants, we dropped this restriction in our system and took benefit of rewriting and rule-based programming controlled by strategies for the generation of so-called primary mechanism. We encoded in **ELAN** eight types of elementary generic primary reactions, that model a special process, specific for high temperatures ($T > 1100K$). Since the detailed mechanisms can involve a huge number of reactions, different types of reduction methods have been applied in the existing software packages. First, we took advantages of conditions in the conditional part of rules to encode the elementary primary generic reactions. Second, we used the power of the **ELAN** strategy language in order to allow the user, the kinetic expert, to activate or to deactivate some classes of reactions. This approach makes our prototype very flexible.

The current prototype was validated for different classes of molecules with the assistance of chemists. Our work has been presented in two international Conferences, ICCS'03 [19] and RTA'03 [18]. Several student projects have contributed this year to the **GasEl** project: Nicolas Hournau, Mathieu Rinck and Belmhel Kassab (UHP) for an interface between **GasEl** and the **THERGAS** system from the chemistry department; Régis Durand (ESIAL) for **ELAN** built-ins for testing equality between molecules; Pierre Henninger (ESIAL) for a graphical interface for **GasEl**.

6.4.3. *A rule language for interaction*

Participants: Carlos Castro, Christophe Ringeissen.

In [32], we propose a rule language for designing interactive component languages and basic coordination languages. In our language, concurrent rules manage interactions and applications of call-back functions on a store of data that can freely be structured as an array, a list, a set of communication channels, etc. This rule language is a kind of abstract machine to write interactive component languages. We then propose a specific component language to write solver cooperation languages. We illustrate the use of this language to specify and implement the basic operators introduced by Castro and Monfroy in a previous work for solving constraints via collaboration of solvers.

This is a joint work with Eric Monfroy from IRIN, University of Nantes.

6.4.4. Rule-based algorithms for combining Nelson-Oppen theories and Shostak theories

Participants: Hélène Kirchner, Christophe Ringeissen, Duc Tran.

In the early 80's, two combination procedures were independently presented by Shostak and by Nelson-Oppen, in order to solve the satisfiability problem in unions of theories. These procedures are applied to program verification when programs involve a mixture of classical data-structures such as lists, arrays, and numbers. These two combination approaches have been opposed for a long time. On one hand, the Nelson-Oppen approach is considered to be quite natural and easy to prove, even if it does not directly yield an efficient implementation. On the other hand, the Shostak approach is difficult to understand, and quite difficult to prove, even if it is implemented in many proving/verification systems because of its good performances. Several recent papers aim at debugging the original Shostak combination procedure by reusing the same error-prone pseudo-code presentation. Our approach is quite different. Instead of "patching" the Shostak approach, we try to "specialize" the general Nelson-Oppen combination schema using the key ideas introduced by Shostak in his combination procedure. In [37], our goal is to reformulate the two approaches in a common framework, and to show that the ideas behind the Shostak algorithm allow us to improve the general principles described by Nelson-Oppen. In order to ease correctness and completeness proofs, we made the choice of using a rule-based formalism to specify our combination procedures. Thus, the proposed combination schemata are given in an abstract way which make them easily understandable and easy to prove. Several classes of theories are considered, such as *Nelson-Oppen* theories and *Shostak* theories, and we study how to combine theories from different classes.

This is a joint work with Silvio Ranise from the Cassis INRIA project.

6.5. Mechanized deduction

Key words: *deduction modulo*, *decision procedures*, *constrained theories*, *induction proofs*, *equational proofs*, *completion procedures*, *termination*, *observable properties*.

We are working on integrating first-order and higher-order deduction, calculi, decision procedures and constraint solving. We also develop proof search procedures for program properties.

On the one hand, this year, we have have designed a general proof-theoretic setting where completion-like processes can be modeled and studied; we have described a superposition calculus where quantifiers are eliminated lazily. We also have studied further the links between proof search, representation and check through the deduction modulo and the ρ-calculus.

On the other hand, we have developed new techniques for property proofs of rewriting with strategies, and especially, we have studied termination of rewriting with strategies.

6.5.1. Abstract canonical presentations and saturation

Participants: Claude Kirchner, Jürgen Stuber.

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. This has been published, in collaboration with Nachum Dershowitz of Tel Aviv university, in [22].

In collaboration with Harald Ganzinger from MPI at Saarbrücken, we have described a superposition calculus where quantifiers are eliminated lazily. Superposition and simplification inferences may employ equivalences that have arbitrary formulas at their smaller side. A closely related calculus is implemented in the Saturate system and has been shown useful on many examples, in particular in set theory. The paper [25] presents a completeness proof and reports on practical experience obtained with the Saturate system.

6.5.2. *Proof search and proof check for equational and inductive theorems*

Participants: Eric Deplagne, Claude Kirchner, Hélène Kirchner, Anamaria Martins Moreira, Fabrice Nahon, Quang-Huy Nguyen.

Although formalized since the beginning of mechanized deduction, the three concepts of proof search, proof representation and proof check are always of fundamental interest, both from the theoretical point of view and for their applications, in particular concerning computer security and dependability. We have considered the theoretical and practical issues of combining rewriting based automated theorem proving and user-guided proof development, with the strong constraint of safe cooperation of both. In practice, we instantiated the theoretical study on the **Coq** proof assistant and the **ELAN** rewriting based system.

A first approach for the cooperation between **Coq** and **ELAN** has been considered, where **Coq** delegates equational proofs by rewriting to **ELAN** that builds a proof term, while doing the proof. This proof term is then returned to **Coq** and checked. In the context of proof by structural induction performed by **Coq**, this technique is currently experimented for proving the base case and the induction case by rewriting. However proofs by Noetherian induction performed by rewriting are much more powerful than structural induction. A second approach, which is yet on-going work, is based on the description at the proof theoretical level of proof by Noetherian induction provided by the deduction modulo framework. We can show how to use narrowing in order to perform proof search for an inductive proof by rewriting, and give hints to build a proof term that can be checked by **Coq**. In order to check the proof, additional proof obligations to justify the Noetherian induction principle in **Coq** are also needed. This has been presented in [21].

Since we are using the Noetherian induction principle at the object level, i.e. as part of the hypothesis of the sequent to be proved, we are studying under which hypothesis this proposition could be added and in which model this is indeed valid. This allows us to better understand the relationship between Noetherianity, the axiom of separability in ZF, and the Noetherian induction principle.

6.5.3. *Inductive termination proofs*

Participants: Olivier Fissore, Isabelle Gnaedig, Hélène Kirchner.

The study of our proof termination method based on induction has been continued this year. For a given term rewriting system, we establish on the ground term algebra that every term t terminates i.e. t has only finite derivations, by explicit induction on the termination property. 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 schematizes normalization of subterms the induction hypothesis is applied on, and narrowing, schematizing rewriting of the resulting form 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.

This year we have focused in particular on the outermost case. We have finished to formalize the constraint mechanism, that is more difficult to handle than for the innermost case and the local strategies, developed previously. In fact, to simulate the outermost rewriting relation with both processes of abstraction and narrowing, we had to introduce a special variable renaming mechanism to deal with all possible redexes of the ground instances of the studied terms. We then have given inference rules to express our termination proof process in this case, and established the correctness of the process [12].

Then, we have worked on automatizing our proof method, in order to improve the efficiency of our induction based termination proof tool CARIBOO and proposed several results. First, we have proposed new conditions under which the ordering constraints, the induction lies on, can be solved. This increases the applicability domain of our method, in particular to well-covered rewriting systems.

We also have developed a dual control mechanism of our inductive process, consisting in testing the unsatisfiability of abstraction constraints instead of verifying its satisfiability, by using decidable sufficient conditions. These new features, as the conditions for solving ordering constraints, have been implemented in the last version of CARIBOO, we have presented at the 6th Workshop on Termination, where we were invited [23].

The thesis of Olivier Fissore, defended in December 2003, presents most of the aspects of our study of termination proof techniques for rewriting under strategies in the context of programming: extensions of the initial inductive method for the innermost strategy, we had proposed in [52], declination to the outermost strategy, to local strategies on operators, to the weak termination proof, and presentation of the simplification technique for combinator-based strategies presented below [10].

6.5.4. Termination proofs for strategies combining rules

Participants: Olivier Fissore, Isabelle Gnaedig, Hélène Kirchner.

We have also studied the termination problem of strategies used in languages like ASF+SDF, Maude, Cafe-OBJ, Stratego, or **ELAN**, where rule application is expressed with combinators. On such strategies, the usual termination proof tools of rewriting cannot work because of the combinators. For the same reason, our inductive proof approach cannot be used either. So, we have proposed a new technique, lying on the analysis and simplification of rule combinators. The idea is to transform the termination problem with such a strategy, very difficult to tackle directly, into the termination study of a conditional rewriting system. The transformation we propose is a simplification in the sense that some combinators are suppressed. The structure of the resulting program is then simpler; it has in general significantly less rules than the initial one, and so is in general more readable. Our simplification can also be seen as an assistance tool for writing rule-based programs since, in addition, the transformations preserve the semantics of programs [24].

6.5.5. A toolbox for verification and proofs of rule-based program properties

Participants: Issam Chebbi, Isabelle Gnaedig.

Our work on the toolbox for verification and proof of rule-based program properties, for which a prototype had been developed last year, has been continued. The prototype, already providing a completeness verification tool, a reducibility test and several termination proof tools as the subterm ordering and the procedures provided by CARIBOO has been completed. The embedding ordering has been implemented, as well as interfaces for the ordering constraint solver CIME [50] and the termination prover APROVE [51].

As our goal is to provide an accessible tool, avoiding to the user expert knowledge, we also have studied in a finer way how the choice between the different proof tools in the toolbox can be automated. When several tools have to be tried, for example several orderings for a termination proof, we have also studied how we could optimize and automate the order these tools are applied. We then have proposed a strategy for using termination tools, lying on the intuitive principle that the simplest one has to be tried first. A use strategy for simplification orderings has been deduced, based on a filtering process: the rewrite system is first studied with the embedding, and all orientable rules are discarded. Then, the remaining rules are oriented with a more powerful simplification ordering, as an RPO or a polynomial ordering, generated by ordering constraint solving. As any simplification ordering contains the embedding, termination of the remaining rules implies termination of the whole system.

We have guessed with complexity arguments, and observed experiments on a large set of examples, that this filtering principle increases the efficiency of the termination proof, by solving less ordering constraints than in the classical way, and by avoiding precedences that would be generated otherwise. It even enables the constraint solvers - once applied on the remaining rules - to conclude in cases where they could not find a solution on the whole system. This filtering principle has been implemented in our toolbox [33].

6.6. Computability and complexity

Key words: *complexity*, *computability*, *implicit complexity*, *analog computations*.

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 a machine independent characterization of elementary computable functions over reals that extends the characterization obtained by Campagnolo. The previous known characterization was restricted to functions over the reals preserving integers.

6.6.1. 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.

Paulin de Naurois' PhD thesis, jointly supervised by Olivier Bournez and Jean-Yves Marion (Calligramme research team), is focused on studying complexity in this model, and especially the links between this model of computation and implicit complexity. Implicit complexity is one approach of the computational complexity of problems, which gives a simple formal setting for studying computational problems with no explicit mention of any resource needed in the computation process. It allows a better understanding of the links between logic, especially proof theory, algorithmic complexity and programming theory.

We have obtained a characterization of BSS computable functions in terms of recursive and primitive recursive functions. Generalizing the characterization of classical polynomial time functions found in [40], we have obtained a characterization of BSS polynomial time functions in terms of safe recursive functions. Later on, we have characterized parallel polynomial time BSS functions in terms of safe recursive functions with substitutions. This result generalizes those found in [58] to the setting of arbitrary structures. These three results have been published in [16]. Extending the results found in [39], we have obtained a precise characterization of every level of the polynomial hierarchy and of the digital polynomial hierarchy in [17].

6.6.2. Analog computations

Participants: Olivier Bournez, Emmanuel Hainry.

There are more than one way to model analog computers. Unifying those models would therefore be a crucial matter as opposed to the discrete case, there is no property saying that those models are equivalent. It is possible to distinguish two groups in those 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 [61]. The main representative of the second group are the real computable functions and a subclass of this set: the set of elementary computable functions.

Some analog computing models seem far too powerful. For example, some make it possible to compute any function in constant time. In such cases, defining a notion of complexity would be impossible. Hence, it is necessary to limit those models in a reasonable domain, namely functions that are known to be physically feasible such as the function contained in the set M_0 defined in [61].

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. We have followed recent works as [46], [45] and [53] which use restricted definitions to keep function with a realistic power. From the classes defined by

Campagnolo, we have built a new class of functions that uses a kind of limits and have proved that this class is exactly the set of real elementary computable functions restricted to $\mathcal{C}^1$ functions.

7. Contracts and Grants with Industry

Glossary

RNTL	National Network on Software Technology
RNRT	National Network for Research on Telecommunication

7.1.1. *Averroes*

Participants: Frédéric Blanqui, Olivier Bournez, Horatiu Cirstea, Claude Kirchner, Huy Nguyen, Anamaria Martins Moreira.

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 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, concerning applications, in Lot 3, concerning the graphical animation of execution traces in the platform developed by the project, in Lot 4, concerning modeling technologies for probabilistic and stochastic systems, and for guaranteeing complexity bounds on consumption of resources, and in Lot 5, about adding rewriting to Coq, and about the mechanization of deduction.

7.1.2. *GasEl*

Participants: Hélène Kirchner, Olivier Bournez, Liliana Ibănescu.

The objectives of the **GasEl** project are to use rewriting and rule-based programming controlled by strategies for a specific problem in physical chemistry: the automated generation of combustion reaction kinetic mechanisms. The partners in this project are INPL, CNRS and **Peugeot Citroën Automobiles**. The laboratories implied are the LORIA and the Department of Chemistry and Physics of Reactions (Nancy).

For computer scientists, the scientific objectives are to use the rewriting systems and rule-based programming controlled by strategies for a very challenging problem, design a new software package, provide a prototype for the core system and to validate it.

For chemists, the scientific objectives are to extend their knowledge about detailed mechanisms for hydrocarbons molecules with cycles, use the new software package to elaborate and validate detailed oxidation and combustion mechanisms for cyclic hydrocarbon molecules.

For the industrial partner, the scientific objectives is to use the detailed mechanisms for hydrocarbons molecules with cycles in order to change the fuel composition and review the conception/design of engines.

7.1.3. *Manifico*

Participants: Horatiu Cirstea, Claude Kirchner, Pierre-Etienne Moreau, Christophe Ringeissen.

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 the 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.

The RNTL Manifico project started on September, 1st. Most of 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 Protheo group, we have developed several compilation algorithms based on the syntactic structure of the rules. This approach allows us to generate specialized data structures to efficiently apply a set of rules. A first objective of this project consists in understanding the formal relationship between these two approaches: the classical term-based indexing algorithm and the RETE algorithm.

A second objective of this project consists in developing a formal basis for integrating rewriting and constraint solving. This 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.

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

8. Other Grants and Activities

8.1. Introduction

Glossary

CPER	Planning Contract between the Government and the Region
PRST	Pole of Scientific and Technological Research of the CPER
PRST-IL	PRST devoted to Software Intelligence
QSL	PRST-IL project on Quality and Safety of Software
Calligramme	INRIA project, Nancy
Cassis	INRIA project, Nancy and Besançon
GDR	CNRS Research Group
GDR-ALP	GDR on Algorithmics, Languages and Programming
FNS	National Fund for Science
ACI	FNS Concerted and Incentive Action
ACI-SI	ACI on Computer and Software Security
ESPRIT	Information Technologies Program of the European Union
Compulog	ESPRIT network on Computational Logic
ERCIM	European Research Consortium for Informatics and Mathematics

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, and with the VALDA project on the automated verification of software in cooperation with Cassis.

8.2.1. Toundra

Participants: Olivier Fissore, Isabelle Gnaedig, Claude Kirchner, Hélène Kirchner.

Key words: *integrated toolbox*, *rule based languages*, *program properties*, *property proofs*, *termination*, *strategy*.

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 the possible relations, they can have one with each other, 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: Horatiu Cirstea, Isabelle Gnaedig, Claude Kirchner, Pierre-Etienne Moreau.

Key words: *asserted software*, *automated provers*, *cooperation*, *proof search*, *rewriting calculus*, *calculus of constructions*.

Modulogic is an ACI-SI 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 action Miro, as well as the FOC group (SPI-LIP6, Paris 6, CEDRIC, CNAM, and Moscova INRIA project) are involved in this project.

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 aimed at interacting with existing tools and languages (like OCAML, 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 languages 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 in 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).

8.5. International bilateral initiatives

AirCube. We are involved in an INRIA associated team called “AirCube” (Rewrite Rule Research). The Aircube project is a strong cooperation between the Protheo group and the Generic Language Technology group[1] of Paul Klint and Mark van den Brand at CWI in Amsterdam. It is based on a close relationship of the scientific interests of both groups and on the common goal to make rewriting concepts and tools widely available for program logic, semantics, and transformation. The two teams share a strong core of common knowledge and know-how in the field of rewriting and their collaboration benefits from complementary point of views on concepts, implementation, applications, and potential transfers. In addition to the construction of a shared knowledge of people and scientific as well as technological information, the scientific outcomes of the AirCube project are presented in Section 6.2.

FERUS. Since 2001, we have a franco-brazilian cooperation with the department of applied mathematics of the Federal University in Natal (UFRN). The aim of this project, called FERUS and supported by CNPq and INRIA, is to work on the design of a reusable software components manager, which could be used for both the Common Algebraic Specification Language and a rule-based language like **ELAN**. An article on the manipulation of algebraic specifications with term and graph representations has been accepted to the Journal of Logic and Algebraic Programming. The final workshop of the FERUS project was held in Natal in October 2003, where the french side was represented by Claude Kirchner, Hélène Kirchner and Christophe Ringeissen.

Chili. Since 2002, we have a franco-chilean cooperation with the Federico Santa Maria Technical University of Valparaiso (UTFSM). This project called COCARS and supported by CONICYT and INRIA, is about the use of rules and strategies for the design of constraint solvers. Christophe Ringeissen made two visits in Valparaiso, one in April 2003, and another one in October/November 2003. In the context of this project, we prepared a paper [32] which was presented at the Joint Annual ERCIM/CoLogNet Workshop on Constraint and Logic Programming.

Hong Kong. Paulin de Naurois carries out his PhD thesis in co-supervision with the Hong Kong *City University*. His supervisors are Felipe Cucker (Hong Kong City University), Olivier Bournez and Jean-Yves Marion (Calligramme project). The PhD subject relies on the complexity in the Blum Shub and Smale model, and its connection with implicit complexity.

Udine. Clara Bertolissi and Alberto Ciaffaglione carry out their PhD thesis in co-supervision with the University of Udine. The supervisors are Furio Honsell at Udine (Italy) and Claude Kirchner, Luigi Liquori and Horatiu Cirstea at Nancy.

8.6. Visiting scientists

- In the context of our french-brazilian cooperation, Anamaria Martins Moreira, UFRN (Native, Brazil), carried out one sabbatical year in PROTHEO until March 2003. Moreover, we accommodated a Brazilian trainee, Anderson Santana, for a four months stay finishing in February 2003.

[1] http://www.cwi.nl/projects/MetaEnv

- Our french-chilean cooperation allows regular visits of Carlos Castro of the Technical University Federico Santa Maria, Valparaiso. In addition, we received the visit of Maria-Cristina Riff, professor at UTSFM, in July 2003 and of Michael Mossens, a student of Carlos Castro, in September 2003.

8.7. Invited lecturers

- David Deharbe and Silvio Ranise (INRIA project Cassis), "L'outil haRVey".
- Christelle Scharff (Pace University, Manhattan), "Direct combination of completion and congruence closure".
- Julien Forest (LRI, Orsay), "Réécriture d'ordre supérieur avec motifs".
- Rachid Echahed (Leibniz, IMAG, Grenoble), "Assuring Secrecy for Concurrent Declarative Programs".
- Pierre Courtieu (INRIA project Lemme), "Jakarta : un environnement pour la certification de la plateforme JavaCard".
- Tayssir Touili (LIAFA, Paris 7), "Reachability analysis of Process Rewrite Systems: Application to the verification of multithreaded recursive programs".
- Luigi Liquori (INRIA project Miro), "An imperative rewriting calculus".
- Frédéric Blanqui (École Polytechnique), "Rewriting modulo in Deduction modulo".
- Florent Kirchner (INRIA project LogiCal), "Coq tacticals and PVS strategies: a small step semantics".
- Virgile Prevosto (LIP6, Paris), "Hériter des spécifications et des théorèmes - Une présentation du système FoC".

9. Dissemination

9.1. Leadership within scientific community

Glossary

IEHSC International Embedded and Hybrid Systems Conference

JFPLC French-speaking Conference on Logic and Constraint Programming

RTA International Conference on Rewriting Techniques and Applications

LPAR Intern. Conference on Logic for Programming Artificial Intelligence and Reasoning

IFIP International Federation for Information Processing

IFIP-WG IFIP Working Group

LICS International Conference on Logics in Computer Science

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

CSL Conference of the European Association for Computer Science Logic

IJCAI International Joint Conference on Artificial Intelligence

PPDP International Conference on Principles and Practice of Declarative Programming

WRLA International Workshop on Rewriting Logic and its Applications

RULE International Workshop on Rule-Based Programming

CP International Conference on Principles and Practice of Constraint Programming

CoSolv Workshop on Cooperative Solvers

PDPAR Workshop on Pragmatics of Decision Procedures in Automated Reasoning

WRS Workshop on Rewriting Strategies

CADE International Conference on Automated Deduction

- Olivier Bournez:
  - Member of the UHP recruitment committee (section 27).
  - Member of the scientific committee of IEHSC'05.
- Isabelle Gnaedig:
  - Manager of the QSL Toundra project.
  - Member of the coordination board of the QSL theme.
  - Member of the program committee of the JFPLC'03.
- Claude Kirchner:
  - Chair of the scientific committee for the national ACI-SI program.
  - Director of education through research at INRIA.
  - Chair of the LORIA building extension committee.
  - Member of the editorial boards of *Journal of Automated Reasoning*, *Journal of Symbolic Computation* (until 31/08), *Journal of Logic Programming*, *Journal of Information Science and Engineering*.
  - Member of the program committees of RTA'03 and of LPAR'03.
  - Chair of the IFIP-WG 1.6 on rewriting and applications.
  - Member of the QPQ, LICS and PPDP advisory boards.
  - Member of the recruitment committees (section 27) of INPL and the University of Metz.
- Hélène Kirchner:
  - Director of LORIA and INRIA Lorraine.
  - Coordinator of the PRST on Software Intelligence.
  - Member of the editorial boards of *Annals of Mathematics and Artificial Intelligence* and *Computing and Informatics*.
  - Member of the editorial board of the QPQ forum on rewriting.
  - Member of the IFIP-WG 1.3 on foundations of systems specifications and the IFIP-WG 1.6 on term rewriting.
  - Member of the management committee of RTP MicroRobotique.
  - Member of the orientation committee of RNTL.
  - Nominee member of the recruitment committee of UHP, Nancy2 and the University Louis Pasteur at Strasbourg.

  - Nominee member of the evaluation commissions of the LRI (Orsay), LSV (Cachan, chairperson), LIFC (Besançon) and Leibniz (Grenoble) laboratories.
  - Member of the program committees of CSL'03 and IJCAI'03.
  - Member of the steering committees of PPDP and RTA.
  - Member of the scientific directorate of the Dagstuhl international conference center.
  - Member and chair of admission committees for technical and administrative staff (ITA) at INRIA.
- Pierre-Etienne Moreau:
  - Member of the program committee of WRLA'03.
  - Co-chair of RULE'03.
  - Member of the LORIA committee for computer and software equipments.
- Christophe Ringeissen:
  - Correspondent for the european projects at LORIA.
  - Member of the program committees of CoSolv'03 and PDPAR'03.

9.2. Teaching

- Olivier Bournez:
  - DEA course on algorithmic verification at UHP.
  - DEA course on computational complexity at UHP.
  - Supervised practical works on algorithmics of parallel and distributed systems to ESIAL 2nd year students.
- 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 "Ecole des Mines".
- Claude Kirchner:
  - DEA course on logic and automated theorem proving with Adam Cichon to various universities.
  - 4 hours introductory lecture on rewriting at the University of Rio Grande Norte, Natal (Brazil).
- Pierre-Etienne Moreau:
  - Supervised practical works on compilation to ESIAL 2nd year students.
- Hélène Kirchner:
  - 15 hours lecture at ENSI (Tunis engineering school) on computation and deduction.
- The project has also teaching assistants and lecturers who teach in various universities of the region.

9.3. Invited talks

- Olivier Bournez:
  - “Some characterizations of complexity classes over arbitrary structures”, VERIMAG (Grenoble) and LIAFA (Paris).
- Claude Kirchner:
  - “The rewriting calculus and the semantic web”, 3rd Franco-Sino Workshop on Web Technologies, Tamkang University, Tamshui (Taiwan).
  - “Rewrite strategies in the rewriting calculus”, WRS’03, Valencia (Spain).
  - “Deduction Modulo”, Workshop on Method for Modalities (M4M), Nancy.
- Hélène Kirchner:
  - “Proof search and proof check for equational and inductive theorems”, CADE-19, Miami.

9.4. Visits

- Claude Kirchner:
  - One week at the computer science laboratory of Natal University (Brazil) in the Ferus project.
- Hélène Kirchner:
  - One week at the computer science laboratory of Natal University (Brazil) in the Ferus project.
- Pierre-Etienne Moreau:
  - One week at the CWI, Amsterdam, in the associated team AirCube.
- Christophe Ringeissen:
  - One week at the department of applied mathematics of Natal University (Brazil).
  - Four weeks at the computer science department of the Technical University Federico Santa Maria in Valparaiso (Chile).

9.5. Thesis and admission committees

- Frédéric Blanqui:
  - PhD thesis SPECIF award committee (SPECIF is a french society of professors and researchers in computer science).
- Isabelle Gnaedig:

   - Admission committee ESIAL.
- Claude Kirchner:
   - Joost Visser, “Generic traversal over typed source code representations”, PhD thesis, University of Amsterdam.
   - Alberto Ciaffaglione, “Certified reasoning on real numbers and objects in co-inductive type theory”, PhD thesis, University of Udine (Italy) and INPL.
   - Marc Aiguier, “Habilitation à diriger des recherches”.
   - Virgile Prevosto, “Conception et implantation du langage FOC pour le développement de logiciels certifiés”, PhD thesis, University Paris 6.
   - Julien Forest, “Réécriture d’ordre supérieur avec motifs”, PhD thesis, University Paris XIII.
   - Jean-Yves Moyen, “Analyse de la complexité et transformation de programmes”, PhD thesis, University Nancy 2.
- Hélène Kirchner:
   - Santiago Escobar, “Strategies and analysis techniques for functional program optimization”, PhD thesis, Universidad Politecnica de Valencia.
   - Frédéric Saubion, “Logique, métaheuristiques et contraintes”, “habilitation à diriger des recherches”, University of Angers.
   - Olivier Fissore, “Terminaison de la réécriture sous stratégies”, PhD thesis, UHP, Nancy.

10. Bibliography

Major publications by the team in recent years

[1] F. BLANQUI. *Definitions by Rewriting in the Calculus of Constructions.* in « Mathematical Structures in Computer Science », Apr, 2003, to appear.

[2] V. BLONDEL, O. BOURNEZ, P. KOIRAN, J. TSITSIKLIS. *The stability of saturated linear dynamical systems is undecidable.* in « Journal of Computer and System Science », 2000.

[3] C. CASTRO. *Building Constraint Satisfaction Problem Solvers Using Rewrite Rules and Strategies.* in « Fundamenta Informaticae », number 3, volume 34, 1998, pages 263-293.

[4] H. CIRSTEA, C. KIRCHNER. *The rewriting calculus - Part I and II.* in « Logic Journal of the Interest Group in Pure and Applied Logics », number 3, volume 9, May, 2001, pages 427-498.

[5] G. DOWEK, T. HARDIN, C. KIRCHNER. *Theorem Proving Modulo.* in « Journal of Automated Reasoning », number 1, volume 31, Nov, 2003, pages 33-72.

[6] C. KIRCHNER, H. KIRCHNER, M. VITTEK. *Designing Constraint Logic Programming Languages using Computational Systems.* V. SARASWAT, P. VAN HENTENRYCK, editors, in « Principles and Practice of Constraint Programming. The Newport Papers », The MIT Press, 1995, chapter 1, pages 131-158.

[7] 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 », 2000.

[8] H. KIRCHNER, C. RINGEISSEN. *Combining Symbolic Constraint Solvers on Algebraic Domains.* in « J. Symbolic Computation », number 2, volume 18, August, 1994, pages 113-155.

[9] C. KIRCHNER, C. RINGEISSEN. *Rule-Based Constraint Programming.* in « Fundamenta Informaticae », volume 34, 1998, pages 225-262.

Doctoral dissertations and "Habilitation" theses

[10] O. FISSORE. *Terminaison de la réécriture sous stratégies.* Thèse d'université, UHP Nancy I, Dec, 2003.

Articles in referred journals and book chapters

[11] H. CIRSTEA, C. KIRCHNER, L. LIQUORI, B. WACK. *Rewrite Strategies in the Rewriting Calculus.* in « Electronic Notes in Theoretical Computer Science », number 4, volume 86, Jun, 2003.

[12] O. FISSORE, I. GNAEDIG, H. KIRCHNER. *Outermost ground termination.* in « Electronic Notes in Theoretical Computer Science », volume 71, Sep, 2003.

[13] C. TINELLI, C. RINGEISSEN. *Unions of Non-Disjoint Theories and Combinations of Satisfiability Procedures.* in « Theoretical Computer Science », number 1, volume 290, Jan, 2003, pages 291-353.

Publications in Conferences and Workshops

[14] G. BARTHE, H. CIRSTEA, C. KIRCHNER, L. LIQUORI. *Pure Patterns Type Systems.* in « Principles of Programming Languages - POPL'2003, New Orleans, USA », ACM, Jan, 2003, http://www.loria.fr/publications/2003/A03-R-002/A03-R-002.ps.

[15] C. BERTOLISSI, H. CIRSTEA, C. KIRCHNER. *Translating Combinatory Reduction Systems into the Rewriting Calculus.* in « 4th International Workshop on Rule-Based Programming (RULE 2003), Valencia, Spain », Jun, 2003, http://www.loria.fr/publications/2003/A03-R-057/A03-R-057.ps, Long version.

[16] O. BOURNEZ, F. CUCKER, P. JACOBÉ DE NAUROIS, J.-Y. MARION. *Computability over an Arbitrary Structure. Sequential and Parallel Polynomial Time.* in « Foundations of Software Science and Computation Structures - FOSSACS'03, Warsaw, Poland », series Lecture Notes in Computer Science, volume 2620, Springer, A. D. GORDON, editor, pages 185-199, Apr, 2003.

[17] O. BOURNEZ, F. CUCKER, P. JACOBÉ DE NAUROIS, J.-Y. MARION. *Safe Recursion Over an Arbitrary Structure : PAR, PH and DPH.* in « Fifth International Workshop on Implicit Computational Complexity - ICC'2003, Ottawa, Canada », series Electronic Notes in Computer Science, number 1, volume 90, Jun, 2003.

[18] O. BOURNEZ, G.-M. CÔME, V. CONRAUD, H. KIRCHNER, M.-L. IBANESCU. *A Rule-Based Approach for Automated Generation of Kinetic Chemical Mechanisms.* in « 14th International Conference on Rewriting Techniques and Applications - RTA'2003, Valencia, Spain », series Lecture Notes in Computer Science,

volume 2706, R. NIEUWENHUIS, editor, pages 30-45, Jun, 2003.

[19] O. BOURNEZ, G.-M. CÔME, V. CONRAUD, H. KIRCHNER, M.-L. IBANESCU. *Automated Generation of Kinetic Chemical Mechanisms Using Rewriting.* in « International Conference on Computational Science - ICCS 2003, Melbourne, Australie », series Lecture Notes in Computer Science, volume 2659, Springer Verlag, A. V. B. PETER M.A. SLOOT, editor, pages 367-376, Jun, 2003.

[20] O. BOURNEZ, M. HOYRUP. *Rewriting Logic and Probabilities.* in « 14th International Conference on Rewriting Techniques and Applications - RTA'2003, Valencia, Spain », series Lecture Notes in Computer Science, volume 2706, Springer, R. NIEUWENHUIS, editor, pages 61-75, Jun, 2003.

[21] E. DEPLAGNE, C. KIRCHNER, H. KIRCHNER, Q. H. NGUYEN. *Proof Search and Proof Check for Equational and Inductive Theorems.* in « Conference on Automated Deduction - CADE-19, Miami, USA », Jul, 2003, http://www.loria.fr/publications/2003/A03-R-464/A03-R-464.ps.

[22] 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.

[23] O. FISSORE, I. GNAEDIG, H. KIRCHNER. *CARIBOO : A Multi-Strategy Termination Proof Tool Based on Induction.* in « 6th International Workshop on Termination 2003 - WST'03, Valencia, Spain », Albert Rubio, pages 77-79, Jun, 2003, http://www.loria.fr/publications/2003/A03-R-436/A03-R-436.ps.

[24] O. FISSORE, I. GNAEDIG, H. KIRCHNER. *Simplification and Termination of Strategies in Rule-Based Languages.* in « Proceedings of the Fifth International Conference on Principles and Practice of Declarative Programming - PPDP'2003, Uppsala, Sweden », pages 124-135, Aug, 2003.

[25] H. GANZINGER, J. STUBER. *Superposition with Equivalence Reasoning and Delayed Clause Normal Form Transformation.* in « Conference on Automated Deduction - CADE-19, Miami Beach, FL, USA », series Lecture notes in Computer Science, volume 2741, Springer, F. BAADER, editor, pages 335-349, Jul, 2003.

[26] A. MARTINS-MOREIRA, C. RINGEISSEN, A. SANTANA. *A Tool Support for Reusing ELAN Rule-Based Components.* in « 4th International Workshop on Rule-Based Programming - RULE'2003, Valence, Espagne », series Electronic Notes in Theoretical Computer Science, number 2, volume 86, Jean-Louis Giavitto, Pierre-Etienne Moreau, Elsevier, P.-E. M. JEAN-LOUIS GIAVITTO, editor, Sep, 2003.

[27] 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 », series Lecture notes in Computer Science, volume 2622, G. HEDIN, editor, pages 61-76, Apr, 2003.

[28] C. RINGEISSEN. *Matching in a Class of Combined Non-Disjoint Theories.* in « 19th International Conference on Automated Deduction - CADE'2003, Miami, Floride, USA », series Lecture Notes in Artificial Intelligence, volume 2741, Geoff Sutcliffe, Springer-Verlag, F. BAADER, editor, pages 212-227, Aug, 2003.

[29] J. STUBER, M. VAN DEN BRAND. *Extracting Mathematical Semantics from LaTeX Documents.* in « Workshop on Principles and Practice of Semantic Web Reasoning - PPSWR'2003, Mumbai, India », Dec, 2003, http://www.loria.fr/publications/2003/A03-R-457/A03-R-457.ps.

[30] M. VAN DEN BRAND, P.-E. MOREAU, J. VINJU. *Environments for Term Rewriting Engines for Free!*. in « International Conference on Rewriting Techniques and Applications - RTA'2003, Valence, Espagne », series Lecture notes in Computer Science, volume 2706, Springer, R. NIEUWENHUIS, editor, pages 424-435, Jun, 2003.

Internal Reports

[31] G. BUREL. *Logique Equationnelle et probabilités selon Halpern.* Stage Magistère 1ière Année, ENS-Lyon, Sep, 2003, http://www.loria.fr/publications/2003/A03-R-321/A03-R-321.ps.

[32] C. CASTRO, E. MONFROY, C. RINGEISSEN. *A Rule Language for Interaction.* Rapport de recherche, Jun, 2003, presented at Joint Annual ERCIM/CoLogNet Workshop on Constraint and Logic Programming, Budapest, Hungary..

[33] I. CHEBBI. *Aide à la preuve et à la validation en programmation par règles.* Stage de DEA, Université Henri Poincaré, Septembre, 2003.

[34] G. FAURE. *Calcul de réécriture explicite.* Stage de DEA, Sep, 2003.

[35] S. HION. *Etude d'un bus logiciel basé sur la réécriture.* Stage de DEA, Jun, 2003.

[36] A. REILLES. *Réécriture dans un cadre concurrent.* Stage de DEA, INPL, Jul, 2003, http://www.loria.fr/publications/2003/A03-R-383/A03-R-383.ps.

[37] D.-K. TRAN. *Coopération de procédures de décision : étude et implantation.* Stage de DEA, Jul, 2003, http://www.loria.fr/publications/2003/A03-R-351/A03-R-351.ps, Le stage s'est fait en collaboration avec Silvio Ranise de l'équipe Cassis..

Bibliography in notes

[38] F. BAADER, T. NIPKOW. *Term Rewriting and all That.* Cambridge University Press, 1998.

[39] S. BELLANTONI. *Predicative recursion and the polytime hierarchy.* P. CLOTE, J. REMMEL, editors, in « Feasible Mathematics II », series *Perspectives in Computer Science*, Birkhäuser, 1994.

[40] S. BELLANTONI, S. COOK. *A new recursion-theoretic characterization of the poly-time functions.* in « Computational Complexity », volume 2, 1992, pages 97-110.

[41] J. A. BERGSTRA, P. KLINT. *The discrete time ToolBus — a software coordination architecture.* in « Science of Computer Programming », number 2–3, volume 31, 1998, pages 205–229, http://citeseer.nj.nec.com/bergstra98discrete.html.

[42] J. A. BERGSTRA, J. KLOP. *Process algebra for synchronous communication.* in « Information and Control », volume 60, 1984, pages 82-95.

[43] P. BOROVANSKÝ, C. KIRCHNER, H. KIRCHNER, C. RINGEISSEN. *Rewriting with strategies in **ELAN**: a functional semantics.* in « International Journal of Foundations of Computer Science », number 1, volume 12,

February, 2001, pages 69-98.

[44] P. BOROVANSKY, C. KIRCHNER, H. KIRCHNER, P.-E. MOREAU. *ELAN from a rewriting logic point of view.* in « Theoretical Computer Science », number 285, July, 2002, pages 155-185.

[45] M. L. CAMPAGNOLO. *Computational Complexity of Real Valued Recursive Functions and Analog Circuits.* Ph. D. Thesis, Universidade Técnica de Lisboa, 2001.

[46] M. L. CAMPAGNOLO, C. MOORE. *Upper and Lower Bounds on Continuous-Time Computation.* in « Unconventional Models of Computation », UMC2K, DMTCS, Springer-Verlag, I. ANTONIOU, C. CALUDE, M. DINEEN, editors, pages 135-153, 2001.

[47] H. CIRSTEA, C. KIRCHNER, L. LIQUORI. *The Rho Cube.* in « 4th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'2001) », series Lecture Notes in Computer Science, volume 2030, Springer, F. HONSELL, M. MICULAN, editors, pages 168-183, Genova (Italy), April, 2001.

[48] H. CIRSTEA, C. KIRCHNER, L. LIQUORI, B. WACK. *The rho cube : some results, some problems.* in « First International Workshop on Higher-Order Rewriting - HOR'02, Copenhague, Danemark », D. Kesner, T. Nipkow & F. van Raamsdonk, July, 2002, Held in conjunction with FLOC'02.

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

[50] E. CONTEJEAN, C. MARCHÉ, B. MONATE, X. URBAIN. *CiME version 2.* 2000, http://cime.lri.fr/.

[51] J. GIESL, R. THIEMANN, P. SCHNEIDER-KAMP, S. FALKE. *The Termination Prover AProVE.* in « Proceedings of the 4th International Workshop on the Implementation of Logics (WIL'03) », Almaty, Kazakhstan, 2003, http://www-i2.informatik.rwth-aachen.de/AProVE/.

[52] I. GNAEDIG, H. KIRCHNER, T. GENET. *Induction for Termination.* Technical report, number 99.R.338, LORIA, Nancy (France), December, 1999.

[53] D. D. S. GRAA. *The General Purpose Analog Computer and Recursive Functions over the Reals.* Technical report, Universidade Técnica de Lisboa, 2002.

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

[55] 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 », number 3, volume 11, Mar, 2001, pages 207-251.

[56] J. KLOP, V. VAN OOSTROM, F. VAN RAAMSDONK. *Combinatory reduction systems: introduction and survey.* in « Theoretical Computer Science », volume 121, 1993, pages 279-308.

[57] D. E. KNUTH, P. B. BENDIX. *Simple word problems in universal algebras.* J. LEECH, editor, in « Computational Problems in Abstract Algebra », Pergamon Press, Oxford, 1970, pages 263-297.

[58] D. LEIVANT, J.-Y. MARION. *Ramified recurrence and computational complexity II: substitution and poly-space.* in « Computer Science Logic, 8th Workshop, CSL '94 », series Lecture Notes in Computer Science, volume 933, Springer, L. PACHOLSKI, J. TIURYN, editors, pages 486-500, Kazimierz,Poland, 1995.

[59] W. MCCUNE. *Solution of the Robbins Problem.* in « JAR », number 3, volume 19, 1997, pages 263-276.

[60] J. MESEGUER. *Conditional rewriting logic as a unified model of concurrency.* in « TCS », number 1, volume 96, 1992, pages 73-155.

[61] C. MOORE. *Recursion Theory on the Reals and Continuous-time Computation.* in « Theoretical Computer Science », volume 162, 1996, pages 23-44.

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

[63] A. VAN DEURSEN. *An Overview of ASF+SDF.* in « Language Prototyping », World Scientific, pages 1-31, 1996, ISBN 981-02-2732-9.