INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team Pareo

Formal Islands: Foundations and Applications

Nancy - Grand Est

THEME SYM

Table of contents

1. Team

Research Scientist

Frédéric Blanqui [CR INRIA, until August 31]
Yves Guiraud [CR INRIA]
Pierre-Etienne Moreau [Team Leader, CR INRIA, HdR]
Claude Kirchner [DR INRIA, Bordeaux, HdR]
Hélène Kirchner [DR CNRS, Bordeaux, HdR]

Faculty Member

Horatiu Cirstea [MC Nancy2]

Technical Staff

Jean-Christophe Bach [Ingénieur Jeune Diplômé INRIA since November 1]

PhD Student

Oana Andrei [CORDI until October 31]
Emilie Balland [MESR until August 31, ATER since September 1]
Tony Bourdier [CORDI since March 1]
Paul Brauner [MESR]
Guillaume Burel [MESR]
Clément Houtmann [MESR]
Radu Kopetz [CORDI until September 15]
Cody Roux [CORDI]
Anderson Santana [Brazil until Mars 31]
Claudia Tavares [Brazil]

Post-Doctoral Fellow

Yohan Boichut [INRIA until August 31]
Richard Bonichon [INRIA until August 31]
Florent Garnier [Nancy 2 half-time ATER until August 31]

Administrative Assistant

Chantal Llorens [Project Assistant]

Other

Pierre Caserta [Master Nancy 1 from February 4 to June 30]
Johan Grande [ENS from June 9 to July 18]
Yassine Guebbas [INPL until February 27]
Laura Lowenthal [CORDI until February 15]
Aurélien Monot [INPL until February 27]

2. Overall Objectives

2.1. Overall Objectives

The PAREO team aims at designing and implementing tools for the specification, analysis and verification of software and systems. At the heart of our project is therefore the will to study fundamental aspects of programming languages (logic, semantics, algorithmic, *etc.*) and to make major contributions to the design of new programming languages. An important part of our research effort will be dedicated to the design of new fundamental concepts and tools to analyze existing programs and systems. To achieve this goal we focus on:

- the improvement of theoretical foundations of rewriting and deduction;
- the integration of the corresponding formal methods in programming and verification environments;
- the practical applications of the proposed formalisms.

2.2. Highlights

- Industrialization by Business Object/SAP of a product developed in *Tom*.
- HDR defense of Pierre-Etienne Moreau on June 13.
- Creation of the Pareo INRIA team on January 1.

3. Scientific Foundations

3.1. Introduction

It is a common claim that rewriting is ubiquitous in computer science and mathematical logic. And indeed the rewriting concept appears from very theoretical settings to very practical implementations. Some extreme examples are the mail system under Unix that uses rules in order to rewrite mail addresses in canonical forms (see the `/etc/sendmail.cf` file in the configuration of the mail system) and the transition rules describing the behaviors of tree automata. Rewriting is used in semantics in order to describe the meaning of programming languages [73] as well as in program transformations like, for example, re-engineering of Cobol programs [84]. It is used in order to compute [56], implicitly or explicitly as in Mathematica, MuPAD or OBJ [64], but also to perform deduction when describing by inference rules a logic [63], a theorem prover [71] or a constraint solver [72]. It is of course central in systems making the notion of rule an explicit and first class object, like expert systems, programming languages based on equational logic [81], algebraic specifications (*e.g.* OBJ), functional programming (*e.g.* ML) and transition systems (*e.g.* Murphi).

In this context, the study of the theoretical foundations of rewriting have to be continued and effective rewrite based tools should be developed. The extensions of first-order rewriting with higher-order and higher-dimension features are hot topics and these research directions naturally encompass the study of the rewriting calculus, of polygraphs and of their interaction. The usefulness of these concepts becomes more clear when they are implemented and a considerable effort is thus put nowadays in the development of expressive and efficient rewrite based programming languages.

3.2. Rule based programming languages

Keywords: *Term rewriting*, *expressiveness*, *programming*, *rule*.

Programming languages are formalisms used to describe programs, applications, or software which aim to be executed on a given hardware. In principle, any Turing complete language is sufficient to describe the computations we want to perform. However, in practice the choice of the programming language is important because it helps to be effective and to improve the quality of the software. For instance, a web application is rarely developed using a Turing machine or assembly language. By choosing an adequate formalism, it becomes easier to reason about the program, to analyze, certify, transform, optimize, or compile it. The choice of the programming language also has an impact on the quality of the software. By providing high-level constructs as well as static verifications, like typing, we can have an impact on the software design, allowing more expressiveness, more modularity, and a better reuse of code. This also improves the productivity of the programmer, and contributes to reducing the presence of errors.

The quality of a programming language depends on two main factors. First, the *intrinsic design*, which describes the programming model, the data model, the features provided by the language, as well as the semantics of the constructs. The second factor is the programmer and the application which is targeted. A language is not necessarily good for a given application if the concepts of the application domain cannot be easily manipulated. Similarly, it may not be good for a given person if the constructs provided by the language are not correctly understood by the programmer.

In the *Pareo* group we target a population of programmers interested in improving the long-term maintainability and the quality of their software, as well as their efficiency in implementing complex algorithms. Our privileged domain of application is large since it concerns the development of *transformations*. This ranges from the transformation of textual or structured documents such as XML, to the analysis and the transformation of programs and models. This also includes the development of tools such as theorem provers, proof assistants, or model checkers, where the transformations of proofs and the transitions between states play a crucial role. In that context, the *expressiveness* of the programming language is important. Indeed, complex encodings into low level data structures should be avoided, in contrast to high level notions such as abstract types and transformation rules that should be provided.

It is now well established that the notion of *term* and *rewrite rule* are two universal abstractions well suited to model tree based data types and the transformations that can be done upon them. Over the last ten years we have developed a strong experience in designing and programming with rule based languages [74], [51], [46]. We have introduced and studied the notion of *strategy* [50], which is a way to control how the rules should be applied. This provides the separation which is essential to isolate the logic and to make the rules reusable in different contexts.

To improve the quality of programs, it is also essential to have a clear description of their intended behaviors. For that, the *semantics* of the programming language should be formally specified.

There is still a lot of progress to be done in these directions. In particular, rule based programming can be made even more expressive by extending the existing matching algorithms to context-matching or to new data structures such as graphs or polygraphs. New algorithms and implementation techniques have to be found to improve the efficiency and make the rule based programming approach effective on large problems. Separating the rules from the control is very important. This is done by introducing a language for describing strategies. We still have to invent new formalisms and new strategy primitives which are both expressive enough and theoretically well grounded. A challenge is to find a good strategy language we can reason about, to prove termination properties for instance.

On the static analysis side, new formalized typing algorithms are needed to properly integrate rule based programming into already existing host languages such as Java. The notion of traversal strategy merits to be better studied in order to become more flexible and still provide a guarantee that the result of a transformation is correctly typed.

3.3. Rewriting calculus

Keywords: *Patterns*, *matching*, *rewriting*, *strategies*.

The huge diversity of the rewriting concept is obvious and when one wants to focus on the underlying notions, it becomes quickly clear that several technical points should be settled. For example, what kind of objects are rewritten? Terms, graphs, strings, sets, multisets, others? Once we have established this, what is a rewrite rule? What is a left-hand side, a right-hand side, a condition, a context? And then, what is the effect of a rule application? This leads immediately to defining more technical concepts like variables in bound or free situations, substitutions and substitution application, matching, replacement; all notions being specific to the kind of objects that have to be rewritten. Once this is solved one has to understand the meaning of the application of a set of rules on (classes of) objects. And last but not least, depending on the intended use of rewriting, one would like to define an induced relation, or a logic, or a calculus.

In this very general picture, we have introduced a calculus whose main design concept is to make all the basic ingredients of rewriting explicit objects, in particular the notions of rule *application* and *result*. We concentrate on *term* rewriting, we introduce a very general notion of rewrite rule and we make the rule application and result explicit concepts. These are the basic ingredients of the *rewriting*- or ρ-calculus whose originality comes from the fact that terms, rules, rule application and application strategies are all treated at the object level (a rule can be applied on a rule for instance).

The λ-calculus is usually put forward as the abstract computational model underlying functional programming. However, modern functional programming languages have pattern-matching features which cannot be directly expressed in the λ-calculus. To palliate this problem, pattern-calculi [82], [76], [70] have been introduced. The rewriting calculus is also a pattern calculus that combines the expressiveness of pure functional calculi and algebraic term rewriting. This calculus is designed and used for logical and semantical purposes. It could be equipped with powerful type systems and used for expressing the semantics of rule based as well as object oriented languages. It allows one to naturally express exception handling mechanisms and elaborated rewriting strategies. It can be also extended with imperative features and cyclic data structures.

The study of the rewriting calculus turns out to be extremely successful in terms of fundamental results and of applications. Different instances of this calculus together with their corresponding type systems have been proposed and studied. The expressive power of this calculus was illustrated by comparing it with similar formalisms [45], [59] and in particular by giving a typed encoding of standard strategies used in first-order rewriting and classical rewrite based languages like *ELAN* and *Tom*.

3.4. Polygraphs and n-categories

Keywords: *Polygraph*, *algebraic topology*, *complexity*, *n-category*, *program analysis*, *termination*.

An *n-category* is an algebraic object where elements, called cells, can be seen as topological objects of dimension up to n. These cells can be glued in n different ways, one for every dimension. *Polygraphs* are presentations by generators and relations of n-categories.

As an example, 3-polygraphs, the ones we encounter the most in both mathematics and computer science, can be seen and manipulated as rewriting systems acting on algebraic circuits. For instance, the following 3-polygraph corresponds to a rule-based computation of the list-splitting function $[x_1, x_2, x_3, \cdots] \mapsto [x_1, x_3, \cdots], [x_2, x_4, \cdots]$ used in the merge-sort algorithm:

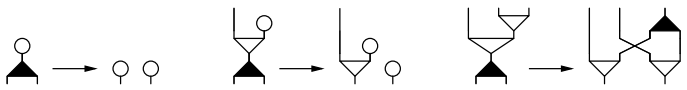

Figure 1.

Following seminal work by Albert Burroni [53] and Yves Lafont [77], it has been proved that many kinds of rewriting systems are low-dimensional polygraphs: abstract rewriting systems are exactly 1-polygraphs; word rewriting systems are exactly 2-polygraphs with one 0-cell; term rewriting systems and, in particular, first-order functional programs, are 3-polygraphs with one 0-cell and with special 2-cells and 3-cells that internalize operations on pointers [65], [66], [69]. Polygraphs also include rewriting-flavoured objects like Petri nets [68] and formal proofs of propositional calculus and of linear logic [67]

Moreover, the n-categories generated by those polygraphs correspond exactly to the reduction space associated to the rewriting system. This leads to the main research direction we explore: one can study the computational properties of a given rewriting system by analysing the topological properties of the corresponding n-category.

Following this direction, *derivations* of 2-categories provide new tools for proving termination of rewriting systems [65], [66]. Moreover, in the case of first-order functional programs, they yield complexity bounds and, as a consequence, polygraphic characterisations of usual complexity classes have been obtained [49][29].

4. Application Domains

4.1. Application Domains

Keywords: *Software*, *quality*, *rule-based programming*.

Beside the theoretical transfer that can be performed via the cooperations or the scientific publications, an important part of the research done in the *Pareo* group team is published within software. *Tom* is our flagship implementation. It is available via the Inria Gforge (http://gforge.inria.fr) and is one of the most visited and downloaded projects. The integration of high-level constructs in a widely used programming language such as Java may have an impact in the following areas:

- Teaching: when (for good or bad reasons) functional programming is not taught nor used, *Tom* is an interesting alternative to exemplify the notions of abstract data type and pattern-matching in a Java object oriented course.
- Software quality: it is now well established that functional languages such as Caml are very successful to produce high-assurance software as well as tools used for software certification. In the same vein, *Tom* is very well suited to develop, in Java, tools such as provers, model checkers, or static analyzers.
- Symbolic transformation: the use of formal anchors makes possible the transformation of low-level data structures such as C structures or arrays, using a high-level formalism, namely pattern matching, including associative matching. *Tom* is therefore a natural choice each time a symbolic transformation has to be implemented in C or Java for instance. *Tom* has been successfully used to implement the Rodin simplifier, for the B formal method.
- Prototyping: by providing abstract data types, private types, pattern matching, rules and strategies, *Tom* allows the development of quite complex prototypes in a short time. When using Java as the host-language, the full runtime library can be used. Combined with the constructs provided by *Tom*, such as strategies, this procures a tremendous advantage.

One of the most successful transfer is certainly the use of *Tom* made by Business Objects/SAP. Indeed, after benchmarking several other rule based languages, they decided to choose *Tom* to implement a part of their software that will be commercialized in 2010. *Tom* is used both in Paris and Vancouver. The standard representation provided by *Tom* is used as an exchange format by the teams of these two sites.

5. Software

5.1. Introduction

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

5.2. ATerm

Keywords: *Tree*, *maximal sharing*, *term*, *xml*.

Participant: Pierre-Etienne Moreau [correspondant].

ATerm (short for Annotated Term) is an abstract data type designed for the exchange of tree-like data structures between distributed applications.

The ATerm library forms a comprehensive procedural interface which enables creation and manipulation of ATerms in C and Java. The ATerm implementation is based on maximal subterm sharing and automatic garbage collection.

A binary exchange format for the concise representation of ATerms (sharing preserved) allows the fast exchange of ATerms between applications. In a typical application—parse trees which contain considerable redundant information—less than 2 bytes are needed to represent a node in memory, and less than 2 bits are needed to represent it in binary format. The implementation of ATerms scales up to the manipulation of ATerms in the giga-byte range.

The ATerm library provides a comprehensive interface in C and Java to handle the annotated term data-type in an efficient manner.

We are involved (with the CWI) in the implementation of the Java version, as well as in the garbage collector of the C version. The Java version of the ATerm library is used in particular by *Tom*.

The ATerm library is documented, maintained, and available at http://www.meta-environment.org/Meta-Environment/ATerms.

5.3. Tom

Keywords: *Pattern matching*, *compilation*, *rule-based programming*, *strategy*.

Participants: Jean-Christophe Bach, Emilie Balland, Paul Brauner, Radu Kopetz, Pierre-Etienne Moreau [correspondant], Claudia Tavares.

Since 2002, we have developed a new system called *Tom* [80], presented in [38] [46]. This system consists of a pattern matching compiler which is particularly well-suited for programming various transformations on trees/terms and XML documents. Its design follows our experiences on the efficient compilation of rule-based systems [75]. 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 with 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.

In addition to the notion of *rule*, *Tom* offers a sophisticated way of controlling their application: a strategy langage. Based on a clear semantics, this language allows to define classical traversal strategies such a *innermost*, *outermost*, *etc.*

Recently, we have developed an extension of pattern matching, called *anti-pattern matching*. This correspond to a natural way to specify *complements* (*i.e.*what should not be there to fire a rule). *Tom* also supports the definition of cyclic graph data-structures, as well as matching algorithm and rewriting rules for term-graphs.

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

5.4. Lemuridae

Keywords: *Superdeduction*, *deduction modulo*, *proof extraction*, *typechecking*.

Participants: Paul Brauner, Guillaume Burel, Clément Houtmann.

Lemuridae is a proof assistant for the sequent calculus instance of superdeduction modulo. It is written in *Tom* and features automatic super-rules derivation with support for axiom directed focussing, automated derivation of induction principles using deduction modulo encoding of higher order logic, as well as some basic automatic tactics. The soundness is ensured by a tiny kernel checking the generated proof trees.

It has been used as a prototyping environment for the developpement of the encoding of Pure Type Systems as well as the simulation of inductive types in superdeduction modulo.

We recently developped [4] a proof-term language for the system which will eventually allow us to share proof witnesses with other provers and began to port the whole system to this new format.

Lemuridae is available in the *Tom* subversion repository, under `applications/lemuridae`.

5.5. CoLoR and Rainbow

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

Participant: Frédéric Blanqui [correspondant].

CoLoR is a *Coq* [55] library on rewriting and termination. It is intended to serve as a basis for certifying the output of automated termination provers like TPA, AProVE, Torpa, etc. It contains libraries on:

- Mathematical structures: relations, semi-rings.
- Data structures: lists, vectors, integer polynomials with multiple variables, finite multisets, matrices.
- Term structures: strings, algebraic terms with symbols of fixed arity, algebraic terms with varyadic symbols, simply typed lambda-terms.
- Transformation techniques: conversion from strings to algebraic terms, conversion from algebraic to varyadic terms, arguments filtering, rule elimination, dependency pairs.
- Termination criteria: polynomial interpretations, multiset ordering, lexicographic ordering, first and higher order recursive path ordering, matrix interpretations, dependency graph decomposition.

Rainbow is a tool for automatically certifying termination proofs expressed in some termination proof grammar (TPG). Termination proofs are translated and checked in *Coq* by using the *CoLoR* library. The termination proof grammar is under development with various participants of the annual international competition on termination[1].

CoLoR and *Rainbow* are distributed under CeCILL license on http://color.loria.fr/.

5.6. Moca

Keywords: *Non-free data types*, *completion*, *functional programming*, *rewriting*.
Participants: Frédéric Blanqui [correspondant], Richard Bonichon, Laura Lowenthal.

Moca is a general construction functions generator for OCaml [57] data types with invariants.

Moca allows the high-level definition and automatic management of complex invariants for data types. In addition, *Moca* provides the automatic generation of maximally shared values, independantly or in conjunction with the declared invariants.

A relational data type is a concrete data type that declares invariants or relations that are verified by its constructors. For each relational data type definition, *Moca* compiles a set of construction functions that implements the declared relations.

Moca supports two kinds of relations:

- algebraic relations (such as associativity or commutativity of a binary constructor),
- general rewrite rules that map some pattern of constructors and variables to some arbitrary user's define expression.

Algebraic relations are primitive, so that *Moca* ensures the correctness of their treatment. By contrast, the general rewrite rules are under the programmer's responsability, so that the desired properties must be verified by a programmer's proof before compilation (including for completeness, termination, and confluence of the resulting term rewriting system).

Algebraic invariants are specified by using keywords denoting equational theories like commutativity and associativity. *Moca* generates construction functions that allow each equivalence class to be uniquely represented by their canonical value.

Moca is distributed under QPL on http://moca.inria.fr/.

6. New Results

6.1. Improvement of theoretical foundations

6.1.1. Term and graph rewriting

Participants: Oana Andrei, Emilie Balland, Pierre Caserta, Horatiu Cirstea, Hélène Kirchner.

[1] http://termination-portal.org/wiki/Termination_Competition

We develop a biochemical calculus [11] based on port graph rewriting for describing molecules, reaction patterns and biochemical network generation. This calculus is an extension of the chemical model by considering structured objects. Then we obtain a natural specification of concurrency and of controlling mechanisms by expressing rewrite strategies as objects of the calculus. We introduce the structure of port graphs and we show how the principles of the biochemical calculus instantiated for port graphs are expressive enough for modeling autonomous systems [18], [22] and biochemical interactions [21]. In addition, strategic rewriting techniques open the way to reason about the computations and to verify properties of the modeled systems [11].

Term-graph rewriting corresponds to an extension of term rewriting to deal with terms that can contain cycles and shared subterms. Based on the formalization of paths and referenced terms, we defined [25] referenced term rewriting as a simulation of term-graph rewriting. Since this simulation is completely based on standard first-order terms, the main interest of this approach is to provide a safe and efficient way to represent and transform term-graphs in a purely term rewriting based language.

Different pattern calculi integrate the functional mechanisms from the λ-calculus and the matching capabilities from rewriting. Several approaches are used to obtain the confluence but in practice the proof methods share the same structure and each variation on the way pattern-abstractions are applied needs another proof of confluence. We have proposed [54] a generic confluence proof where the way pattern-abstractions are applied is axiomatized. This approach that handles only the cases where the matching is unitary has been extended [40] to formalisms using non-unitary matching and in particular equational (commutative) matching.

We introduced in [36] the notion of abstract strategies for abstract reduction systems. Adequate properties of termination, confluence and normalization under strategy can then be defined. Thanks to this abstract concept, we draw a parallel between strategies for computation and strategies for deduction. We defined deduction rules as rewrite rules, a deduction step as a rewriting step and a proof construction step as a narrowing step in an adequate abstract reduction system. Computation, deduction and proof search are thus captured in the uniform foundational concept of abstract reduction system in which abstract strategies have a clear formalisation.

6.1.2. Algebraic and topological properties of rewriting systems

Participant: Yves Guiraud.

The property of *finite derivation type* is a homotopical property of rewriting systems. Intuitively, when a rewriting system has finite derivation type, there are only finitely many non-trivial choices one can make in any given computation. A family of such elementary choices is a *homotopy basis* of the rewriting system.

This property has been introduced by Craig Squier [83] for word rewriting systems, see also [79]. It turns out that it is an invariant of the monoid being presented, *i.e.* the monoid whose elements are the connected components of the corresponding reduction graph. Moreover, when a monoid admits a presentation by a finite and convergent word rewriting system, then it has finite derivation type and the critical branchings generate a homotopy basis.

In a joint work with Philippe Malbos (Université Lyon 1) [43], we have generalised the property of finite derivation type from monoids presented by word rewriting systems, *i.e.* 1-categories presented by 2-polygraphs, to n-categories presented by $(n+1)$-polygraphs. We have recovered Squier's results and proved, with the following counter-example, that the existence of a finite convergent presentation was not sufficient enough to guarantee that an n-category has finite derivation type, starting with $n = 2$:

Figure 2.

However, we have identified an extra condition, *finite indexation*, and proved that a 2-category with a presentation by a finite, convergent and finitely indexed 3-polygraph has finite derivation type. Usual 3-polygraphs have this property: in particular, the canonical translation of a left-linear term rewriting system into a 3-polygraph is always finitely indexed; as a consequence, if the Lawvere theory corresponding to a given function on abstract data types has not finite derivation type, then the function cannot be computed by a first-order functional program.

This work also produced an alternative, more intuitive way to formulate and prove Saunder MacLane's coherence theorem for monoidal categories [78].

6.1.3. Mechanized deduction

Participants: Frédéric Blanqui, Paul Brauner, Guillaume Burel, Clément Houtmann, Claude Kirchner, Hélène Kirchner, Cody Roux.

Subtyping has been very well studied in computer science, where a type is a subtype of another type if its elements behave in some sense like those of the supertype. The formalisation of mathematics can be done in certain powerfull type theories, where intuitively types can correspond to the sets of non-formal mathematics. However, the common practice of mathematics rely on a very large number of (resolvable) ambiguities, some of which can not be solved by naive subtyping frameworks. It is therefore of interest for the formalisation of mathematics (and the practice of programming) to try to extend these, and coercive subtyping seems to be the appropriate framework. In [35] we considered the simply typed calculus with a coercive subtyping system where coercions can be given between arbitrary types and not only atomic types. We proved that subtyping is decidable, and that this implies that type checking is decidable in this framework.

Pure Pattern Type Systems combine in a unified setting the frameworks and capabilities of rewriting and λ-calculus. Their type systems, adapted from Barendregt's λ-cube, are specially interesting from a logical point of view. Strong normalization, an essential property for logical soundness, had only been conjectured so far: in [17], together with Benjamin Wack, we have given a positive answer for the simply-typed system and the dependently-typed system, basing our proof on a translation of terms and types from Pure Pattern Type Systems into the λ-calculus.

Superdeduction and deduction modulo are methods specially designed to ease the use of first-order theories in predicate logic. In [44] we have revisited the superdeduction paradigm by comparing it with the focusing approach. In particular we prove a focalization theorem for cut-free superdeduction modulo: we have shown that permutations of inference rules can transform any cut-free proof in deduction modulo into a cut-free proof in superdeduction modulo and conversely, provided that some hypotheses on the synchrony of reasoning axioms are verified. It implies that cut-elimination for deduction modulo and for superdeduction modulo are equivalent. Since several criteria had already been proposed for theories that do not break cut-elimination of the corresponding deduction modulo system, these criteria also imply cut-elimination of the superdeduction modulo system, provided our synchrony hypotheses hold. Finally we have designed a tableaux method for superdeduction modulo which is sound and complete provided cut-elimination holds.

We showed [37] how the superdeduction strong normalization property entails the expected recursive computational behaviour at the proof-term level when encoding inductive types by the type of their elimination scheme. This allowed us use [41] the notion of superconsistency [58], which relates a semantic notion of model for a firt-order theory expressed as equation on types to strong normalization of the associated deduction modulo (hence superdeduction [52]) proof system. This has lead to a concise and semantic argument of Gödel System T strong normalization.

We have also investigated how superdeduction can be used as a logical framework, to be able to emulate other proof systems. In particular, we proved that every functional pure type system can be encoded into supernatural deduction [30]. Our aim is to help proof assistants cooperate.

We have studied further the impact of deduction modulo to proof-length speedups in higher-order arithmetic [42]. We have shown how the higher-order part of an arithmetic proof can be encoded by a very simple rewrite system without increasing the length of proofs. We also have described the whole higher-order arithmetic as a

purely computational theory, that is, we have defined a rewrite system, and pure first-order logic modulo this system is equivalent to higher-order arithmetic, also w.r.t. proof lengths.

We finally showed with Fabrice Nahon how narrowing can be used as an efficient proof-search method for inductive equational proofs by restricting the unification to defined-innermost positions [16]. To this end, we designed the *IndNarrow* semi-decision procedure and showed how to extract a deduction modulo proof out of every successful instance of the algorithm. This has lead us to implement a prototype which proved several complex equalities. We now look forward to exporting the resulting deduction modulo proofs to our proof assistant *lemuridae*.

In [26], we investigate a new version of the Calculus of Inductive Constructions (CIC) on which the proof assistant Coq is based: the Calculus of Congruent Inductive Constructions, which truly extends CIC by building in arbitrary first-order decision procedures. The deduction is still in charge of the CIC kernel, while computation is outsourced to dedicated first-order decision procedures that can be taken from the shelves provided they deliver a proof certificate. The soundness of the whole system becomes an incremental property following from the soundness of the certificate checkers and that of the kernel. A detailed example shows that the resulting style of proofs becomes closer to that of the working mathematician.

Finally, in [19], we briefly survey automated termination proof methods for higher-order calculi. We then concentrate on the higher-order recursive path ordering, for which we provide an improved definition, the Computability Path Ordering. This new definition appears indeed to capture the essence of computability arguments à la Tait and Girard, therefore explaining the name of the improved ordering.

6.1.4. Tree automata

Participant: Yohan Boichut.

Rewriting approximation computation as in [60] has been shown as a well suited approach to perform verification on finite or infinite system: security protocols [61], [48] and Java bytecode programs [47]. This technique is also called regular model-checking based on completion of tree automata. In few words, for a given term rewriting system and a given initial set of terms, we compute an over-approximation of the set of terms actually reachable. Thus, we prove that a given term t is unreachable by verifying that t is not in the computed approximation. However, when the term t is in the approximation, nothing can be said and the user has to propose a new approximation. In [27], we describe a new approach for automatically generating over-approximations guided by the set of unwanted terms and a technique of automatic refinements.

The tree automata completion technique fits only for a certain class of term rewriting systems. Indeed, due to the non-determinism of tree automata involved in computations of approximations, the over-approximations computed may be not sound i.e. not really over-approximations. Of course, a solution to this problem is the determinization of tree automata. This solution is theoretically simple but not efficient in practice. We have proposed in [28] an algorithm based on determinization of tree automata but theoretically more efficient.

We have recently shown in [15], [39] the theoretical limits of this technique in any case. Indeed, the analyses may be inconclusive since for a given term rewriting system and a given initial set of terms, some terms actually unreachable may be in all computable approximations.

6.2. Integration of formal methods in programming languages

6.2.1. Formal islands and Tom

Participants: Emilie Balland, Paul Brauner, Yves Guiraud, Radu Kopetz, Aurélien Monot, Pierre-Etienne Moreau, Claudia Tavares.

In [1] we have proposed a framework which makes possible the integration of formally defined constructs into an existing language. The *Tom* system is an instance of this principle: terms, first-order signatures, rewrite rules, strategies and matching constructs are integrated into Java and C for instance. The high level programming features provided by this approach are presented in [20]. The *Tom* system is documented in [38]. A general overview of the research problem raised by this approach are presented in [13].

One interest of *Tom* is to make the compilation process independent of the considered data-structure. Given any existing data-structure, using a *formal anchor* definition, it becomes possible to match and to apply transformation rules on the considered data-structures. In [12], we have presented a tool that automatically extracts *formal anchors* from an existing data-structure. The interest has been demonstrated on an example based on *JPA: Java Persistance API*.

During the internship of Aurélien Monot, we developed a compiler for polygraphic programs, as defined in [3] and in [29]. It is written in *Tom* and, given a polygraphic program expressed in an XML document, produces a *Tom* program computing the same function. The "terms" of the polygraph are normalised thanks to *Tom* internal strategy language, while the "rules" application uses *Tom* pattern-matching functionality. The manuscript can be found at http://www.loria.fr/~guiraudy/divers/monot.pdf.

We are currently working on the definition of a new type system for *Tom* along with the associated type inference and checking algorithms. This type system allows to declare polymorphic first-order signatures along subtyping and (in)equations, which will eventually extend the expressivity of the *Tom* language by allowing the encoding of BNF grammar. Moreover, it provides a strictly defined semantics to *Tom*'s "star variables" which modelize matched sublists of associative functions symbols.

6.2.2. Extension of pattern-matching

Participants: Emilie Balland, Claude Kirchner, Radu Kopetz, Pierre-Etienne Moreau.

Graphs are omnipresent in program analysis. The implementation of static analysers require the representation of control-flow and data-flow graphs for instance. As *Tom* can only manipulate tree structures, we proposed an extension to deal with graph structures as shown in [25], [24]. The main idea is to use paths to represent cycles and shared parts. This leads to an original and clean way for representing, matching and transforming graphs in a rewrite-based environment.

Negation is intrinsic to human thinking and most of the time when searching for something, we base our patterns on both positive and negative conditions. In [32], the notion of term was extended to the one of anti-term, i.e. terms that may contain complement symbols. We present theoretical algorithms for anti-pattern matching as well as an extension of *Tom* that integrates these notions.

In [31], we generalize the syntactic anti-pattern matching to anti-pattern matching modulo an arbitrary equational theory E, and we study the specific and practically very useful case of associativity, possibly with a unity (AU). To this end, based on the syntacticness of associativity, we present a rule-based associative matching algorithm, and we extend it to AU. This algorithm is then used to solve AU antipattern matching problems. This allows us to be generic enough so that for instance, the AllDiff standard predicate of constraint programming becomes simply expressible in this framework. AU anti-patterns are implemented in the Tom language and we show some examples of their usage.

6.3. Practical applications

6.3.1. Security policy analysis

Participants: Tony Bourdier, Horatiu Cirstea, Yassine Guebbas, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau, Anderson Santana.

We have addressed the problem of authoring and analyzing policies in a modular way using techniques developed in the field of term rewriting, focusing especially on the use of rewriting strategies [14]. Well-established term rewriting techniques allow us to check properties of policies such as the absence of conflicts and the property of always returning a decision. A rich language for expressing rewriting strategies is used to define a theory of modular construction of policies, in which we can better understand the preservation of properties of policies under composition.

The access control mechanisms should guarantee that information can be accessed only by authorized users and thus prevent all information leakage. We proposed [33] a methodology for specifying and implementing access control policies using the rewrite based framework *Tom*. This approach allows us to check that any reachable state obtained following an access granted in the implementation satisfies the policy specification. We show that when security levels are not totally ordered some information leakage can be detected.

We have also described [34] how to perform queries over these rule-based policies in order to increase the trust of the policy author on the correct behavior of the policy. The analysis we provide is founded on the narrowing process, which provides both the necessary abstraction for simulating executions of the policy over access requests and the mechanism for solving what-if queries from the security administrator. We illustrate this general approach by the analysis of a firewall system policy.

6.3.2. Program analysis

Participants: Yohan Boichut, Emilie Balland, Pierre-Etienne Moreau.

One goal of the ANR Ravaj project is to analyze Java programs to know whether a given piece of code is reachable or not. We have used a term based encoding as well as tree automata techniques to compute, given a Java program, a set of reachable states. In [23], using a tree automata completion technique, it has been shown that the non reachability of a term t can be verified by computing an over-approximation of the set of reachable terms and proving that t is not in the over-approximation. Since the verification of real programs gives rise to rewrite models of significant size, efficient implementations of completion are essential. We present in this paper a TRS transformation preserving the reachability analysis by tree automata completion. This transformation makes the completion implementation based on rewriting techniques possible. Thus, the reduction of a term to a state by a tree automaton is fully handled by rewriting. This approach has been prototyped in Tom and the first experiments are very promising relative to the state-of-the-art tool Timbuk [62].

7. Other Grants and Activities

7.1. Regional Initiatives

We obtained a financial support from the Lorraine region for funding the research activities of Oana Andrei and Yohan Boichut.

7.2. National Initiatives

We participate in the “Logic and Complexity” part of the GDR–IM (CNRS Research Group on Mathematical Computer Science), in the projects “Logic, Algebra and Computation” (mixing algebraic and logical systems) and “Geometry of Computation” (using geometrical and topological methods in computer science).

We participate and co-animate the “Transformation” group of the GDR–GPL (CNRS Research Group on Software Engineering).

7.2.1. ARC Quotient (2007-2008)

Participants: Frédéric Blanqui, Richard Bonichon, Laura Lowenthal.

This project gathers people from INRIA Nancy - Grand Est (Frédéric Blanqui, Richard Bonichon, Laura Lowenthal), INRIA Paris - Rocquencourt (Pierre Weis and Damien Doligez), Université Paris 6 (Thérèse Hardin, Renaud Rioboo) and CNAM (David Delahaye, Catherine Dubois). Its aim is to study and certify the use of non-free concrete data types in functional programming and develop an extension of OCaml providing such types.

7.2.2. ANR Complice (2008-2012)

Participant: Yves Guiraud.

The ANR project "Complexité implicite, concurrence et extraction" (Complice), headed by Patrick Baillot (CNRS, LIP Lyon), federates researchers around the topic of implicit computational complexity. The coordinator for the LORIA site is Guillaume Bonfante (Carte). The project will start in December.

7.2.3. ANR Infer (2007-2009)

Participants: Guillaume Burel, Claude Kirchner.

This ANR project is a grouping of three teams through their common interest for a new approach to proof theory, called "deep inference". The project aims at refining its potential and at applying it to problems related to the foundations of logic and to more practical questions in the algorithmic of deductive systems, such as identity of proofs, Curry-Howard isomorphism, complexity of proofs, formulation of "exotic" logical systems, links with other paradigms like deduction modulo, etc. For more information, see the Infer website at http://www.lix.polytechnique.fr/~lutz/orgs/infer.html.

7.2.4. ANR Inval (2005-2008)

Participant: Yves Guiraud.

The ANR project "Invariants algébriques des systèmes informatiques" (Inval), headed by Éric Goubault (CEA Saclay), federates researchers in mathematics and theoretical computer science. Its main objective is to favour the transfer of ideas and methods between both communities. The coordinator for the LORIA site is François Lamarche (Calligramme). For more information, see the Inval website at http://www.pps.jussieu.fr/~inval/index.html. The project terminates on November 30.

7.2.5. ANR Ravaj (2007-2009)

Participants: Emilie Balland, Yohan Boichut, Pierre-Etienne Moreau.

Ravaj (Réécriture et Approximation pour la Vérification d'Applications Java) is an ANR project coordinated by Thomas Genet (Irisa). The goal is to model Java bytecode programs using term rewriting and to use completion techniques to compute the set of reachable terms. Then, it is possible to check some properties related to reachability (in particular safety and security properties) on the modeled system using tree automata intersection algorithms.

7.2.6. ANR SSURF (2007-2009)

Participants: Tony Bourdier, Horatiu Cirstea, Anderson Santana.

"SSURF: Safety and Security under FOCAL" is an ANR project coordinated by Mathieu Jaume (LIP6). The SSURF project consists in characterizing and studying the required features that an Integrated Development Environment (IDE) must provide in order not only to obtain software systems in conformance with high Evaluation Assurance Levels (EAL-5, 6 and 7), but also to ease the evaluation process according to various standards (*e.g.*IEC61508, CC, ...). Moreover we aim at developing a formal generic framework describing various security properties, *e.g.*access control policies, together with their implementations using such an IDE.

7.3. International Initiatives

Chili. We have an associated team "VanaWeb" that started in 2008 and continues the collaboration initiated during the joint project INRIA-CONICYT (Chili), VANANAA (formerly, COCARS). It is a project on rules and strategies for the hybrid resolution of constraint problems with applications to composition problems for the Web. We have many exchanges with Carlos Castro and his group (UTFSM, Valparaiso, Chile).
Brazil. Project INRIA-CNPq (Brazil), DA CAPO - Automated deduction for the verification of specifications and programs. It is a project on the development of proof systems for the verification of specifications and software components. The coordinators of this project are David Déharbe (UFRN Natal, Brazil) and Christophe Ringeissen (CASSIS). On the french side, DA CAPO also involves the CASSIS project.

Japan. We are part of the joint French-Japanese cooperative program on "Foundations of provably secure software technology and its applications" whose goal is to provide the foundations of provably secure software using our logical and mathematical methodology for practical and crucial applications, especially focusing on applications to security of new-generation smart cards.

7.4. Exterior research visitors

- Philippe Malbos, Lyon, one week in April and one week in October.
- Anderson Santana, Brazil, two weeks in October.

7.5. Invited lecturers

The program of the seminars is available at http://pareo.loria.fr/.

- Germain Faure (Barcelona), *SAT modulo la théorie de l'arithmétique linéaire: solvers exacts, inexacts et commerciaux*.
- Piero Bonatti (Napoli), *Semantic web policies for security and privacy*.
- Thérèse Hardin (Paris 6), *A few remarks about developping safety or security critical systems within inductive formal systems*.
- Daniel Dougherty (Worcester), *Alchemy: transmuting specifications into implementations*.
- Barry Jay (Sydney), *Programming with patterns in bondi* and *Typed pattern calculus: beyond the Curry-Howard Isomorphism*.
- Paolo Baldan (Padova), *Verification of graph transformation systems*.
- Dorel Lucanu (Iasi), *Circular coinduction-based techniques for proving behavioral properties*.

8. Dissemination

8.1. Animation of the scientific community

Horatiu Cirstea:

- Program committees of RULE 2008 (International Workshop on Rule-Based Programming), JFLA 2008 (Journées Francophones des Langages Applicatifs).
- Steering committee of RULE.

Yves Guiraud:

- Committee of Nancy INRIA Research Center.

Claude Kirchner:

- Director of the INRIA Bordeaux - Sud-Ouest research center.
- Co-coordinator of the Franco-Japanese 3 years cooperation program on security founded by CNRS and JST.
- Editorial boards of *Journal of Automated Reasoning*, *Journal of Applied Logic*.
- Program committee of LSFA'07, Brazilian Workshop on Logical and Semantic Frameworks, with Applications, Chair of the scientific committee of the second international school on Rewriting (ISR'2007).
- Chair of the IFIP WG 1.6 working group on rewriting and applications.
- Member of the advisory board of LICS.
- Member of the working group on research and perspectives of the CISSI.

- Program committees of RTA'08 (International Conference on Rewriting Techniques and Applications), LPAR'08 (International Conference on Logic for Programming, Artificial Intelligence and Reasoning), honorary president of CRISIS'08 (International Conference on Risks and Security of Internet and Systems), LSFA'08 (Brazilian Workshop on Logical and Semantics Frameworks, with Applications), WRLA'08 (Workshop on Rewriting Logic and its Applications), PPDP'08 (Symposium on Principles and Practice of Declarative Programming).
- Member of the evaluation panel of Luxembourg university.
- President of the evaluation committee of CELAR/SSI.
- Member of the steering committe of the ANR Arpege program.
- Chair of the INRIA working group on ethics.
- President of the DGRI committee on "Etude sur la recherche académique en sécurité informatique." (http://scoulond.insa-lyon.fr/dgri).
- Co-organizer of the French-Japanese workshop on sécurity (Nancy March 13-14 and Tokyo December 5,6)

Hélène Kirchner:

- Deputy scientific director at INRIA.
- Editorial boards of *Annals of Mathematics and Artificial Intelligence* (Associated Editor), *Computing and Informatics* and *Logical Methods in Computer Science*.
- Program committees of AMAST'08, WRS'08.
- Member of the ANR selection committee of the programmes "Non thématique 2007" and "Jeunes chercheurs 2007" in "Sciences et Technologies de l'Information".
- Member of the ANR steering committees "Domaines Emergents" (DEFIS) and "Concepts Systèmes et Outils pour la Sécurité Globale" (CSOG).
- Member of the steering committee of GIS (Groupement d'Intérêt Scientifique) 3SGS (Surveillance, sûreté et sécurité des grands systèmes).

Pierre-Etienne Moreau:

- Program committee of SLE 2008 (International Conference on Software Language Engineering), LFSA 2008 (Brazilian Workshop on Logical and Semantics Frameworks, with Applications), WRS 2008 (Workshop on Reduction Strategies in Rewriting and Programming), WRLA 2008 (Workshop on Rewriting Logic and its Applications)
- Steering committee of LDTA (Workshop on Language Descriptions, Tools and Applications).

8.2. Teaching

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

Frédéric Blanqui:

- 3-days introductory course on logic and typed lambda-calculus at Tsinghua University, Beijing, China (March 2008)

Horatiu Cirstea:

- Master course in Nancy on programming and proving with rule based languages, with Pierre-Etienne Moreau.
- Course on rewriting techniques and transformation at CARI 2008 (9e Colloque Africain sur la Recherche en Informatique et en Mathématiques Appliquées)
- Supervision of Pierre Caserta's Master thesis: "Confluence de calculs à motifs".
- Supervision of Yassine Guebbas's internship (École des Mines de Nancy), with Pierre-Etienne Moreau.

Yves Guiraud:

– Supervision of Aurélien Monot's internship (École des Mines de Nancy), with Pierre-Etienne Moreau.

Claude Kirchner:

– Lecture (five hald days) on "Rewriting - Computation and Deduction " Tsinghua University, Feb.

Pierre-Etienne Moreau:

– Master course in Nancy on programming and proving with rule based languages, with Horatiu Cirstea,
– Supervision of Aurélien Monot's internship (École des Mines de Nancy), with Yves Guiraud,
– Supervision of Yassine Guebbas's internship (École des Mines de Nancy), with Horatiu Cirstea.

Hélène Kirchner:

– Lecture on rewriting techniques and transformation at CARI 2008, (9e Colloque Africain sur la Recherche en Informatique et en Mathématiques Appliquées), Maroc, October 2008.
– Lecture (five hald days) on "Rewriting - Computation and Deduction" at Tsinghua University, Beijing, April 2008.

8.3. Invited talks

Yves Guiraud:

– Université Lyon 1, "Complexity of polygraphic programs" and "Higher-dimensional categories with finite derivation type", September 25 and 26.

Claude Kirchner:

– CISTRANA Trust & Security Seminar "SESUR: Sécurité et Sûreté Informatique" Bruxelles (Janvier);
– TFIT'08 (Taiwanese-French Conference on Information Technology): "Security Policies and Strategic Rewriting" Taipei (March);
– Symposium SFJTI (Société Franco-Japonaise des technologies de l'Information) "Security and Informatics: Problematic and Challenge" Tokyo (March);
– CARI'08: "Customization of Deduction Systems" Rabat (October);
– Colloquium in honor of Hubert Comon: "Weaving computation with deduction" Paris (Novembre);
– The 4th Franco-Japanese Computer Security Workshop: "Antipatterns: a way to say what you don't want" Tokyo (December).

Hélène Kirchner:

– "Computer Security: Software Issues, Problems and Challenges", "A Higher-Order Graph Calculus for Autonomic Computing", Invited talks at the Conference *Graph Theory, Computational Intelligence and Thought*
in Honor of Martin Golumbic, Israel, September 2008.

Pierre-Etienne Moreau:

– "Rule-Based Programming in Java", Workshop on Rule-Based Programming.

8.4. Visits

Yves Guiraud:

– Institut Camille Jordan, Lyon, one week in June and one week in December.

Claude Kirchner

- SRI International, one week in August.

8.5. Theses

- Oana Andrei: “Un calcul de réécriture de graphes : applications à la biologie et aux systèmes autonomes”, PhD.
- Radu Kopetz: “Contraintes d’anti-fltrage et programmation par réécriture”, PhD.
- Pierre-Etienne Moreau: “Programmation et confiance”, HDR.
- Anderson Santana: “Réécriture et modularité pour les politiques de sécurité”, PhD.

8.6. Thesis and admission committees

Frédéric Blanqui:

- Substitute member of the Saint-Etienne University recruitment committee.
- Referee of Enrico Tassi’s PhD thesis on “ Interactive Theorem Provers: issues faced as a user and tackled as a developer” at Bologna University, Italy.

Horatiu Cirstea:

- Member of recruitment committee (section 27) of Nancy2.
- Oana Andrei: “Un calcul de réécriture de graphes: applications à la biologie et aux systèmes autonomes”.
- Mohamed Tafjouti: “Archivage et suivi médical informatisé des patients en exploration fonctionnelle respiratoire pédiatrique”.

Claude Kirchner:

- Radu Kopetz: “Contraintes d’anti-filtrage et programmation par réécriture”, PhD université de Nancy (co-advisor).
- Anderson De Olivera: “Réécriture et Modularité pour les Politiques de Sécurité”, PhD université de Nancy (co-advisor).
- Benjamin Werner: “Faire simple pour pouvoir faire compliqué Contributions à une Théorie des Types pratique” Habilitation à diriger les recherches, Université Paris 11 (Président).
- Pierre-Etienne Moreau “Programmation et confiance” Habilitation à diriger les recherches, université de Nancy
- Myrto Arapinis “Sécurité des protocoles cryptographiques : décidabilité et résultats de réduction” PhD ENS Cachan.

Hélène Kirchner:

- Anderson De Olivera: “Réécriture et Modularité pour les Politiques de Sécurité”, PhD université de Nancy (co-advisor).
- Oana Andrei: “Un calcul de réécriture de graphes: applications à la biologie et aux systèmes autonomes” (advisor).
- Mathieu Jaume: “Descriptions formelles - Application au contrôle d’accès”, Habilitation à diriger les recherches, Université Paris 6 (Rapporteur).
- Member of the UHP recruitment committee (section 27).

Pierre-Etienne Moreau:

- Radu Kopetz: “Contraintes d’anti-filtrage et programmation par réécriture”, PhD (co-advisor)
- Pierre Parrend: “Software Security Models for Service-Oriented Programming (SOP) Platforms”, PhD
- Member of the UHP recruitment committee (section 27).

9. Bibliography

Major publications by the team in recent years

[1] E. BALLAND, C. KIRCHNER, P.-E. MOREAU. *Formal Islands*, in "11th International Conference on Algebraic Methodology and Software Technology, Kuressaare, Estonia", M. JOHNSON, V. VENE (editors), LNCS, vol. 4019, Springer-Verlag, July 2006, p. 51–65, http://www.loria.fr/~moreau/Papers/BallandKM-AMAST2006.pdf.

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

[3] G. BONFANTE, Y. GUIRAUD. *Polygraphic programs and polynomial-time functions*, in "Logical Methods in Computer Science", to appear, 2007.

[4] P. BRAUNER, C. HOUTMANN, C. KIRCHNER. *Principles of Superdeduction*, in "Twenty-Second Annual IEEE Symposium on Logic in Computer Science - LiCS 2007, Wroclaw Pologne", IEEE Computer Society, 2007.

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

[6] N. DERSHOWITZ, C. KIRCHNER. *Abstract canonical presentations*, in "Theoretical Computer Science", vol. 357, n^{o} 1-3, July 2006, p. 53–69.

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

[8] Y. GUIRAUD. *Termination orders for 3-dimensional rewriting*, in "J. Pure and Appl. Algebra", vol. 207, n^{o} 2, October 2006, p. 341-371.

[9] C. KIRCHNER, R. KOPETZ, P.-E. MOREAU. *Anti-Pattern Matching*, in "16th European Symposium on Programming, Braga, Portugal", Lecture Notes in Computer Science, vol. 4421, Springer, 2007, p. 110–124, http://www.loria.fr/~moreau/Papers/KirchnerKM-2007.pdf.

[10] P.-E. MOREAU, C. RINGEISSEN, M. VITTEK. *A Pattern Matching Compiler for Multiple Target Languages*, in "12th Conference on Compiler Construction, Warsaw (Poland)", G. HEDIN (editor), LNCS, vol. 2622, Springer-Verlag, May 2003, p. 61–76, http://www.loria.fr/~moreau/Papers/MoreauRV-CC2003.ps.gz.

Year Publications

Doctoral Dissertations and Habilitation Theses

[11] O. ANDREI. *Un calcul de réécriture de graphes: applications à la biologie et aux systèmes autonomes*, Ph. D. Thesis, Institut National Polytechnique de Lorraine - INPL, 11 2008, http://tel.archives-ouvertes.fr/tel-00337558/en/.

[12] R. KOPETZ. *Contraintes d'anti-filtrage et programmation par réécriture*, Ph. D. Thesis, Institut National Polytechnique de Lorraine - INPL, 10 2008, http://tel.archives-ouvertes.fr/tel-00337539/en/.

[13] P.-E. MOREAU. *Programmation et confiance*, Ph. D. Thesis, Institut National Polytechnique de Lorraine - INPL, 06 2008, http://tel.archives-ouvertes.fr/tel-00337408/en/.

[14] A. SANTANA DE OLIVEIRA. *Réécriture et Modularité pour les Politiques de Sécurité*, Ph. D. Thesis, Université Henri Poincaré - Nancy I, 03 2008, http://tel.archives-ouvertes.fr/tel-00335079/en/.

Articles in International Peer-Reviewed Journal

[15] P.-C. HEAM, Y. BOICHUT. *A Theorerical Limit for Safety Verification Techniques with Regular Fix-point Computations*, in "Information Processing Letters", vol. 108, 2008, p. 1-2, http://hal.inria.fr/inria-00328487/en/.

[16] F. NAHON, C. KIRCHNER, H. KIRCHNER, P. BRAUNER. *Inductive Proof Search Modulo*, in "Special issue of the Annals Of Mathematics And Artificial Intelligence on First- order Theorem Proving (à paraître)", 2008, http://hal.inria.fr/inria-00337380/en/.

[17] B. WACK, C. HOUTMANN. *Strong Normalization in two Pure Pattern Type Systems*, in "Mathematical Structures in Computer Science", vol. 18, 2008, p. 431-465, http://hal.inria.fr/inria-00186815/en/.

Invited Conferences

[18] O. ANDREI, H. KIRCHNER. *Strategic Port Graph Rewriting for Autonomic Computing*, in "The Fourth Taiwanese-French Conference on Information Technology - TFIT'08, Taiwan Taipei", 2008, http://hal.inria.fr/inria-00328491/en/.

[19] F. BLANQUI, J.-P. JOUANNAUD, A. RUBIO. *The computability path ordering: the end of a quest*, in "7th EACSL Annual Conference on Computer Science Logic - CSL'08 LNCS, Italie Bertinoro", vol. 5213, 2008, http://hal.inria.fr/inria-00288209/en/.

[20] P.-E. MOREAU, A. REILLES. *Rules and Strategies in Java*, in "7th International Workshop on Reduction Strategies in Rewriting and Programming - WRS 2007 Proceedings of the 7th International Workshop on Reduction Strategies in Rewriting and Programming (WRS 2007) Electronic Notes in Theoretical Computer Science, France Paris", J. GIESL (editor), vol. 204, Elsevier, 2008, p. 71-82, http://hal.inria.fr/inria-00185698/en/.

International Peer-Reviewed Conference/Proceedings

[21] O. ANDREI, H. KIRCHNER. *A Biochemical Calculus Based on Strategic Graph Rewriting*, in "The Third International Conference on Algebraic Biology - AB'08, Autriche Hagenberg", 2008, http://hal.inria.fr/inria-00328536/en/.

[22] O. ANDREI, H. KIRCHNER. *A Higher-Order Graph Calculus for Autonomic Computing*, in "Graph Theory, Computational Intelligence and Thought. A Conference Celebrating Martin Charles Golumbic's 60th Birthday, Israël Haifa", 2008, http://hal.inria.fr/inria-00328554/en/.

[23] E. BALLAND, Y. BOICHUT, T. GENET, P.-E. MOREAU. *Towards an Efficient Implementation of Tree Automata Completion*, in "12th International Conference on Algebraic Methodology and Software Technology - AMAST'08 Lecture Notes in Computer Science, États-Unis d'Amérique Urbana, Illinois", J. MESEGUER, G. ROSU (editors), vol. 5140, Springer, 2008, p. 67-82, http://hal.inria.fr/inria-00304010/en/.

[24] E. BALLAND, P. BRAUNER. *Term-graph rewriting in Tom using relative positions*, in "4th International Workshop on Computing with Terms and Graphs TERMGRAPH 2007 ENTCS, Portugal Braga", I. MACKIE (editor), vol. 203, ELSEVIER, 2008, p. 3-17, http://hal.inria.fr/inria-00129515/en/.

[25] E. BALLAND, P.-E. MOREAU. *Term-graph rewriting via explicit paths*, in "RTA: International Conference on Rewriting Techniques and Applications RTA Lecture Notes in Computer Science, Autriche Hagenberg", A. VORONKOV (editor), vol. 5117, Springer, 2008, p. 32-47, http://hal.inria.fr/inria-00173535/en/.

[26] F. BLANQUI, J.-P. JOUANNAUD, P.-Y. STRUB. *From formal proofs to mathematical proofs: a safe, incremental way for building in first-order decision procedures*, in "5th IFIP International Conference on Theoretical Computer Science - TCS 2008 IFIP, Italie Milan", vol. 273, 2008, http://hal.inria.fr/inria-00275382/en/.

[27] Y. BOICHUT, R. COURBIS, P.-C. HEAM, O. KOUCHNARENKO. *Finer is better: Abstraction Refinement for Rewriting Approximations*, in "19th International Conference on Rewriting Techniques and Applications - RTA'2008 Lecture Notes in Computer Science, Autriche Hagenberg", A. VORONKOV (editor), vol. 5117, Springer, 2008, p. 48-62, http://hal.inria.fr/inria-00327583/en/.

[28] Y. BOICHUT, R. COURBIS, P.-C. HEAM, O. KOUCHNARENKO. *Handling Left-Quadratic Rules when Completing Tree Automata*, in "2nd Workshop on Reachability Problems - RP'08 Electronic Notes in Theoretical Computer Science, Royaume-Uni Liverpool", V. HALAVA, I. POTAPOV (editors), Elsevier Science Publishers, 2008, http://hal.inria.fr/inria-00329900/en/.

[29] G. BONFANTE, Y. GUIRAUD. *Intensional properties of polygraphs*, in "4th International Workshop on Computing with Terms and Graphs - TERMGRAPH 2007 Electronic Notes in Theoretical Computer Science, Portugal Braga", vol. 203(1):65-77, 2008, http://hal.inria.fr/inria-00129391/en/.

[30] G. BUREL. *A First-Order Representation of Pure Type Systems Using Superdeduction*, in "23rd Annual IEEE Symposium on Logic In Computer Science Logic In Computer Science, États-Unis d'Amérique Pittsburgh, PA", F. PFENNING (editor), IEEE Computer Society, 2008, p. 253-263, http://hal.inria.fr/inria-00198543/en/.

[31] C. KIRCHNER, R. KOPETZ, P.-E. MOREAU. *Anti-Pattern Matching Modulo*, in "Second International Conference on Language and Automata Theory and Applications - LATA 2008 Language and Automata Theory and Applications Lecture Notes in Computer Science, Italie Tarragone", C. MARTÍN-VIDE, F. OTTO, H. FERNAU (editors), vol. 5196, Springer-Verlag, 2008, p. 275-286, http://hal.inria.fr/inria-00337722/en/.

[32] R. KOPETZ, P.-E. MOREAU. *Software Quality Improvement via Pattern Matching*, in "11th International Conference on Fundamental Approaches to Software Engineering - FASE 2008 Fundamental Approaches to Software Engineering Lecture Notes in Computer Science, Hongrie Budapest", J. FIADEIRO, P. INVERARDI (editors), vol. 4961, Springer-Verlag, 2008, p. 296-300, http://hal.inria.fr/inria-00336703/en/.

[33] A. SANTANA DE OLIVEIRA, C. KIRCHNER, H. KIRCHNER. *Analysis of Rewrite-Based Access Control Policies*, in "3rd International Workshop on Security and Rewriting Techniques, États-Unis d'Amérique Pittsburgh", 2008, http://hal.inria.fr/inria-00335088/en/.

[34] A. SANTANA DE OLIVEIRA, P.-E. MOREAU, H. CIRSTEA. *Rewrite Based Specification of Access Control Policies*, in "3rd International Workshop on Security and Rewriting Techniques - SecReT 2008, États-Unis d'Amérique Pittsburgh", 2008, http://hal.inria.fr/inria-00335091/en/.

National Peer-Reviewed Conference/Proceedings

[35] C. ROUX. *Types Simples, Logique et Coercions Implicites*, in "19e Journées Francophones des Langages Applicatifs - JFLA 2008, France Etretat", INRIA, 2008, p. 79-90, http://hal.inria.fr/inria-00202824/en/.

Scientific Books (or Scientific Book chapters)

[36] C. KIRCHNER, F. KIRCHNER, H. KIRCHNER. *Strategic Computations and Deductions*, in "Festchrift in honor of Peter Andrews", Studies in Logic and the Foundations of Mathematics, Elsevier, 2008.

Research Reports

[37] L. ALLALI, P. BRAUNER. *A Semantic Normalization Proof for Inductive Types*, Rapport de recherche, 2008, http://hal.inria.fr/inria-00280410/en/.

[38] E. BALLAND, P. BRAUNER, R. KOPETZ, P.-E. MOREAU, A. REILLES. *Tom Manual*, Rapport Technique, 2008, http://hal.inria.fr/inria-00121885/en/.

[39] Y. BOICHUT, P.-C. HEAM. *A Theoretical Limit for Safety Verification Techniques with Regular Fix-point Computations*, RR-6411, Rapport de recherche, 2008, http://hal.inria.fr/inria-00204579/en/.

[40] P. CASERTA. *Confluence de calcul à motifs*, Rapport de recherche, 2008, http://hal.inria.fr/inria-00337931/en/.

Other Publications

[41] L. ALLALI, P. BRAUNER. *A semantic normalization proof for a system with recursors*, 2008, http://hal.inria.fr/inria-00211877/en/.

[42] G. BUREL. *Efficiently Simulating Higher-Order Arithmetic by a First-Order Theory Modulo*, 2008, http://hal.inria.fr/inria-00278186/en/.

[43] Y. GUIRAUD, P. MALBOS. *Higher-dimensional categories with finite derivation type*, 2008, http://hal.archives-ouvertes.fr/hal-00326974/en/.

[44] C. HOUTMANN. *Axiom directed Focusing*, 2008, http://hal.inria.fr/inria-00212059/en/.

References in notes

[45] M. ABADI, L. CARDELLI. *A Theory of Objects*, Springer Verlag, 1996.

[46] E. BALLAND, P. BRAUNER, R. KOPETZ, P.-E. MOREAU, A. REILLES. *Tom: Piggybacking rewriting on java*, in "18th International Conference on Rewriting Techniques and Applications - (RTA), Paris, France", Lecture Notes in Computer Science, vol. 4533, jun 2007, p. 36-47, http://hal.inria.fr/inria-00142045/en/.

[47] Y. BOICHUT, T. GENET, T. JENSEN, L. L. ROUX. *Rewriting Approximations for Fast Prototyping of Static Analyzers*, in "proceedings of RTA", Lecture Notes in Computer Science 4533, Springer, 2007, p. 48–62.

[48] Y. BOICHUT, P.-C. HÉAM, O. KOUCHNARENKO. *Handling Algebraic Properties in Automatic Analysis of Security Protocols*, in "ICTAC'06", K. BARKAOUI, A. CAVALCANTI, A. CERONE (editors), Lecture Notes in Computer Science, vol. 4281, Springer, 2006, p. 153–167, http://dx.doi.org/10.1007/11921240_11.

[49] G. BONFANTE, Y. GUIRAUD. *Polygraphic programs and polynomial-time functions*, in "Logical Methods in Computer Science", 38 pages, accepted for publication, 2007.

[50] P. BOROVANSKÝ, C. KIRCHNER, H. KIRCHNER. *Controlling Rewriting by Rewriting*, in "Proceedings of the first international workshop on rewriting logic - (WRLA), Asilomar (California)", J. MESEGUER (editor), Electronic Notes in Theoretical Computer Science, vol. 4, September 1996.

[51] P. BOROVANSKÝ, C. KIRCHNER, H. KIRCHNER, P.-E. MOREAU. *ELAN from a rewriting logic point of view*, in "Theoretical Computer Science", vol. 2, n^{o} 285, July 2002, p. 155-185.

[52] P. BRAUNER, G. DOWEK, B. WACK. *Normalization in Supernatural deduction and in Deduction modulo*, 2007, http://www.loria.fr/~brauner/submission_94.pdf.

[53] A. BURRONI. *Higher-dimensional word problems with applications to equational logic*, in "Theoretical Computer Science", vol. 115, n^{o} 1, July 1993, p. 43-62.

[54] H. CIRSTEA, G. FAURE. *Confluence of pattern-based lambda-calculi*, in "Rewriting Techniques and Applications - RTA'07", Lecture Notes in Computer Science, vol. 4533, 2007, p. 78-92.

[55] COQ DEVELOPMENT TEAM. *The Coq Proof Assistant Reference Manual, Version 8.1*, INRIA Rocquencourt, France, 2006, http://coq.inria.fr/.

[56] N. DERSHOWITZ. *Computing with Rewrite Systems*, in "Information and Control", vol. 65, n^{o} 2/3, 1985, p. 122–157.

[57] D. DOLIGEZ, J. GARRIGUE, X. LEROY, D. RÉMY, J. VOUILLON. *The Objective Caml system release 3.10, Documentation and user's manual*, INRIA, France, 2007, http://caml.inria.fr/.

[58] G. DOWEK. *Truth Values Algebras and Proof Normalization*, in "TYPES", 2006, p. 110-124, http://dx.doi.org/10.1007/978-3-540-74464-1_8.

[59] K. FISHER, F. HONSELL, J. C. MITCHELL. *A Lambda Calculus of Objects and Method Specialization*, in "Nordic Journal of Computing", vol. 1, n^{o} 1, 1994, p. 3–37.

[60] T. GENET. *Decidable approximations of sets of descendants and sets of normal forms*, in "proceedings of RTA", Lecture Notes in Computer Science, vol. 1379, Springer-Verlag, 1998.

[61] T. GENET, F. KLAY. *Rewriting for Cryptographic Protocol Verification*, in "proceedings of CADE", Lecture Notes in Computer Science, vol. 1831, Springer-Verlag, 2000, p. 271–290.

[62] T. GENET, V. VIET TRIEM TONG. *Timbuk 2.0 – A Tree Automata Library*, 2000, http://www.irisa.fr/lande/genet/timbuk/, IRISA / Université de Rennes 1.

[63] J.-Y. GIRARD, Y. LAFONT, P. TAYLOR. *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science, vol. 7, Cambridge University Press, 1989.

[64] J. A. GOGUEN, C. KIRCHNER, H. KIRCHNER, A. MÉGRELIS, J. MESEGUER, T. WINKLER. *An Introduction to OBJ-3*, in "Proc. 1st CTRS Workshop, Orsay (France)", J.-P. JOUANNAUD, S. KAPLAN (editors), Lecture Notes in Computer Science, Also as internal report CRIN: 88-R-001, vol. 308, Springer-Verlag, July 1987, p. 258–263.

[65] Y. GUIRAUD. *Présentations d'opérades et systèmes de réécriture*, Ph. D. Thesis, Université Montpellier 2, June 2004.

[66] Y. GUIRAUD. *Termination orders for 3-dimensional rewriting*, in "Journal of Pure and Applied Algebra", vol. 207, n^{o} 2, October 2006, p. 341-371.

[67] Y. GUIRAUD. *The three dimensions of proofs*, in "Annals of Pure and Applied Logic", vol. 141, n^{o} 1-2, August 2006, p. 266-295.

[68] Y. GUIRAUD. *Two polygraphic presentations of Petri nets*, in "Theoretical Computer Science", vol. 360, n^{o} 1-3, August 2006, p. 124-146.

[69] Y. GUIRAUD. *Polygraphs for termination of left-linear term rewriting systems*, Preprint, 2007.

[70] B. JAY, D. KESNER. *Pure Pattern Calculus*, in "European Symposium on Programming – ESOP'06, Vienna, Austria", Lecture Notes in Computer Science, vol. 3924, Springer, March 2006, p. 100-114.

[71] J.-P. JOUANNAUD, H. KIRCHNER. *Completion of a set of rules modulo a set of Equations*, in "SIAM J. of Computing", vol. 15, n^{o} 4, 1986, p. 1155–1194.

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

[73] G. KAHN. *Natural Semantics*, Technical report, n^{o} 601, INRIA Sophia-Antipolis, February 1987.

[74] C. KIRCHNER, H. KIRCHNER, M. VITTEK. *Designing Constraint Logic Programming Languages using Computational Systems*, in "Proc. 2nd CCL Workshop, La Escala (Spain)", F. OREJAS (editor), September 1993.

[75] H. KIRCHNER, P.-E. MOREAU. *Promoting Rewriting to a Programming Language: A Compiler for Non-Deterministic Rewrite Programs in Associative-Commutative Theories*, in "Journal of Functional Programming", vol. 11, n^{o} 2, 2001, p. 207-251, http://www.loria.fr/~moreau/Papers/KirchnerM-JFP2001.pdf.

[76] J. W. KLOP, V. VAN OOSTROM, R. DE VRIJER. *Lambda calculus with patterns*, in "Theor. Comput. Sci.", vol. 398, n^{o} 1-3, 2008, p. 16–31.

[77] Y. LAFONT. *Towards an algebraic theory of boolean circuits*, in "J. Pure and Appl. Algebra", vol. 184, n^{o} 2-3, 2003, p. 257-310.

[78] S. MAC LANE. *Categories for the working mathematician*, 2nd, Springer, 1998.

[79] P. MALBOS. *For string rewriting systems the homotopical and homological finiteness conditions coincide*, preprint, 50 pages, 2007.

[80] P.-E. MOREAU, C. RINGEISSEN, M. VITTEK. *A Pattern Matching Compiler for Multiple Target Languages*, in "12th Conference on Compiler Construction - (CC)", G. HEDIN (editor), Lecture Notes in Computer Science, vol. 2622, Springer-Verlag, May 2003, p. 61–76, http://www.loria.fr/~moreau/Papers/MoreauRV-CC2003.ps.gz.

[81] M. J. O'DONNELL. *Computing in Systems Described by Equations*, Lecture Notes in Computer Science, vol. 58, Springer-Verlag, 1977.

[82] S. PEYTON-JONES. *The implementation of functional programming languages*, Prentice-Hall, 1987.

[83] C. SQUIER. *A finiteness condition for rewriting systems*, in "Theoretical Computer Science", Revised by Friedrich Otto and Yuji Kobayashi, vol. 131, n^{o} 2, 1994, p. 271-294.

[84] M. VAN DEN BRAND, A. VAN DEURSEN, P. KLINT, S. KLUSENER, E. A. VAN DER MEULEN. *Industrial Applications of ASF+SDF*, in "AMAST '96", M. WIRSING, M. NIVAT (editors), Lecture Notes in Computer Science, vol. 1101, Springer-Verlag, 1996, p. 9-18.