INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Pareo

Formal Islands: Foundations and Applications

Nancy - Grand Est

Theme : Programs, Verification and Proofs

Table of contents

1. Team

Research Scientist
Yves Guiraud [CR INRIA]

Faculty Member
Horatiu Cirstea [MC Nancy2]
Pierre-Etienne Moreau [Team Leader, CR INRIA until August 31, PR INPL since September 1, HdR]

External Collaborator
Claude Kirchner [DR INRIA, HdR]
Hélène Kirchner [DR CNRS, HdR]

Technical Staff
Jean-Christophe Bach [Ingénieur Jeune Diplômé INRIA]

PhD Student
Emilie Balland [ATER until August 31, CR INRIA Bordeaux since September 1]
Tony Bourdier [CORDI]
Paul Brauner [MESR, ATER Bordeaux since September 1]
Guillaume Burel [MESR until August 31. Postdoctoral fellow at Max-Planck-Institute für Informatik, Saarbrücken, since September 14]
Clément Houtmann [MESR]
Cody Roux [CORDI]
Claudia Tavares [Brazil]

Administrative Assistant
Chantal Llorens

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.

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 [56] as well as in program transformations like, for example, re-engineering of Cobol programs [71]. It is used in order to compute [44], implicitly or explicitly as in Mathematica, MuPAD or OBJ [47],

but also to perform deduction when describing by inference rules a logic [46], a theorem prover [54] or a constraint solver [55]. 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 [66], 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

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 [58], [42], [39]. We have introduced and studied the notion of *strategy* [41], 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

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 [68], [60], [53] 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 [35], [45] 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

An $(n+1)$-*polygraph* is a presentation of an n-category by generators, called n-cells, and rewriting rules, called $(n+1)$-cells [43]. For example, the list-splitting map $[x_1, x_2, x_3, \cdots] \mapsto [x_1, x_3, \cdots], [x_2, x_4, \cdots]$, used in the merge-sort algorithm, can be presented (and computed) by the convergent 3-polygraph of Figure 1.

Polygraphs are a common algebraic setting for many different types of rewriting systems. Indeed, abstract and word rewriting systems are 1-polygraphs and 2-polygraphs, respectively [43]. More important, every term rewriting system has a canonical translation into a 3-polygraph with similar computational properties (termination, confluence, complexity) and, particularly, in the left-linear case, hence, when the term rewriting system is a first-order functional program [43], [61][6] [51]. Other types of rewriting-flavoured objects also fit in the class of 3-polygraphs, like Petri nets [50], propositional calculus and linear logic [49], Reidemeister moves on braids or tangles diagrams [48].

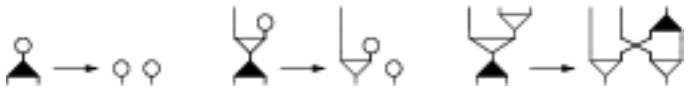

Figure 1. A polygraphic program for the split function

The computational properties of n-polygraphs appear as topological properties of the n-category they generate, that we study with tools adapted from algebraic topology.

For example, in the case of a 3-polygraph, termination and complexity respectively correspond to the finiteness and to the size of the three-dimensional reduction paths. To measure them, we use *derivations* of 2-categories [6]: informally, they associate each 2-cell with a "current propagation map" and a "heat production map". Well-chosen derivation give termination proofs and can also be used to analyse the complexity of polygraphs: a spatial bound is given by "currents" and a temporal one, by "heats". The complexity classes P and NP have characterisations in terms of polygraphs that admit a certain type of derivations [12] [40].

For another example, convergence of a 3-polygraph is linked with the four-dimensional holes created by three-dimensional paths between them. When a polygraph has "good" computational properties, then it has *finite derivation type*, meaning that it has a finite number of generating holes [7]. In particular, every first-order functional program has finite derivation type, linking this topological property to computability.

4. Application Domains

4.1. Application Domains

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. ATerm

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.

5.2. Tom

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

Since 2002, we have developed a new system called *Tom* [65], presented in [28], [39]. 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 [59]. 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.

5.3. Lemuridae

Participants: Paul Brauner [correspondant], 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.

This year, the kernel has been entirely rewritten to take advantage of the proofterms developped in [4]. To this end, we have developped *FreshGom*, an extension of the *Tom* language providing mechanisms that ease the manipulation terms containing binders. This allowed us to export *Lemuridae*'s proofs to other formats, including Parigot's $\lambda\mu$-calculus [67] and *Coq* proofterms [34].

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

5.4. Cat

Participant: Yves Guiraud [correspondant].

Cat is an automatic termination prover for first-order functional programs. It translates such a rewriting system into a 3-polygraph and tries to find a derivation proving its termination, using the results of [6] [51]. If possible, it seeks a derivation that proves that the program is polynomial [3]. For the moment, Cat is a prototype giving results on simple examples. One goal is to specialize it to prove termination of Tom strategies, translated as first-order functional programs.

5.5. Catex

Participant: Yves Guiraud [correspondant].

Catex is a preprocessor for (pdf)Latex for automatic production of diagrams representing 2-cells from their algebraic expression. It follows the same design as Tom, a Catex file being a Latex file enriched with formal islands corresponding to those algebraic expressions, such as:

```
\deftwocell[red]{delta : 1 -> 2}
\deftwocell[orange]{mu : 2 -> 1}
\deftwocell[crossing]{tau : 2 -> 2}
\twocell{(delta *0 delta) *1 (1 *0 tau *0 1) *1 (mu *0 mu)}
```

Catex dissolves such an island into Latex code, using the PGF/Tikz package. Executed on the result, (pdf)Latex produces the diagram presented in Figure 2.

Figure 2. Diagram produced by Catex

Catex is being tested internally, until the syntax is stable enough. It has been used to produce a research paper [30] and slides for talks and reports.

6. New Results

6.1. Improvement of theoretical foundations

6.1.1. Term and graph strategic rewriting

Participants: Emilie Balland, Tony Bourdier, Horatiu Cirstea, Hélène Kirchner.

From our previous work on biochemical applications, the structure of port graph and a rewriting calculus have proved to be well-suited formalisms for modeling interactions between proteins. Port graphs as graphs with multiple edges and loops, with nodes having explicit connection, points, called ports, and edges attaching to ports of nodes. In [36], port graphs have been proposed as a formal model for distributed resources and grid infrastructures, where each resource is modeled by a node with ports. The lack of global information and the autonomous and distributed behavior of components are modeled by a multiset of port graphs and rewrite rules which are applied locally, concurrently, and non-deterministically. Some computations take place wherever it is possible and in parallel, while others may be controlled by strategies. In [18], we then define an abstract biochemical calculus that instantiates to a rewrite calculus on these graphs. Rules and strategies are themselves port graphs, i.e. first-class objects of the calculus. As a consequence, they can be rewritten as well, and rules can create new rules, providing a way of modeling adaptive systems. This approach also provides a formal framework to reason about computations and to verify useful properties. We show how structural properties of a modeled system can be expressed as strategies and checked for satisfiability at each step of the computation. This provides a way to ensure invariant properties of a system. This work is a contribution to the formal specification and verification of adaptive systems and to theoretical foundations of autonomic computing.

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 [10] 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.

In [57], we introduce 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 define 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.

In the continuation of this work, we proposed in [20] an alternative definition of abstract reduction systems and introduced a general definition of abstract strategies which is extensional in the sense that a strategy is defined explicitly as a set of derivations of an abstract reduction system. We introduce also a more intensional definition supporting the abstract view but more operational in the sense that it describes a means for determining such a set. We characterize the class of extensional strategies that can be defined intensionally. We also give some hints towards a logical characterization of intensional strategies.

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 [70]. 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. In a joint work with Philippe Malbos (Université Lyon 1) we have studied this propery for presentations of n-categories by polygraphs.

We have recovered Squier's results on presentations of monoids by word rewriting systems. We have proved, with the counter-example given in Figure 3, 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 3. The 3-polygraph of planar necklaces with pearls

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, an equational theory must have finite derivation type to admit a presentation by a first-order functional program. Finally, this work gives a general setting to express and prove coherence theorems, such as Mac Lane's one for monoidal categories [62], by using simple rewriting methods. This work has been published in [15].

Then we have generalized the notion of *identities among relations* to polygraphs, giving an algebraic interpretation of the previous results. We have proved that the elements of a homotopy basis of a polygraph yield a generating set for its natural system of identities among relations. As a consequence, if a polygraph has finite derivation type, then its identities among relations are finitely generated. This work is contained in [30] and has been presented to the Workshop on Computer Algebra Methods and Commutativity of Algebraic Diagrams, held in Toulouse.

We are currently working on a generalization of Fox differential calculus for n-categories. The objective is to use polygraphs with good computational properties to build small resolutions of n-categories, allowing one to concretely compute the (co)homology groups of a given n-category. This work fits into the project of studying complexity classes of programs, characterized in terms of derivations of 2-categories, by means of cohomological methods, since the first cohomology group of the canonical resolution of any usual algebraic object classifies its derivations.

6.1.3. Mechanized deduction

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

Higher order rewrite systems are useful abstractions to model both operational semantics of programming languages and equational reasoning in certain theories. The termination of these systems is very useful for proving correctness of programs, finding decision procedures, and proving consistency of certain theorem proving systems based on type theory.

A well studied method of proving termination is *size based termination* which allows comparison of sizes of arguments in recursive calls by typing. However the semantics of such systems is unclear, as is their relation to other techniques of termination. in [19], together with Frederic Blanqui, we show that a certain kind of algebraic semantics can be given to such typing systems, and that termination can indeed be shown as a special case of *higher-order semantic labelling*, combined with a simple precedence argument. This gives a uniform termination proof to a large number of possible size-based termination systems.

We have investigated how, starting from an axiomatic presentation of a theory, it is possible to present it by means of rewrite rules so that it can be used in deduction modulo. To ensure good proof-theoretical properties as well as the completeness of the proof-search procedures based on deduction modulo, the resulting rewrite system must be so that the cut rule is admissible in deduction modulo. In classical logic, this can always be done, first by transforming the axioms into a rewrite system and then by completing this rewrite system using a Knuth-Bendix like procedure to recover the cut admissibility [13]. However, in intuitionistic logic, there are theories which cannot be transformed into a rewrite system with the cut admissibility. We show that it even undecidable to know if it is possible. Nonetheless, by interleaving the transformation of axioms into rewrite

rules and the cut-admissibility-recovering completion, we can propose a (non-terminating, possibly failing) procedure that works on a large class of theories [22].

These results show how deduction modulo can be used to get better automation in proofs. In general, we investigated how deduction modulo and superdeduction provides better proofs, by studying three simplicity criteria: cut admissibility, proof length, and expressiveness [11].

In [17], we present an original narrowing-based proof search method for inductive theorems in equational rewrite theories given by a rewrite system $\mathcal{R}$ and a set E of equalities. It has the specificity to be grounded on deduction modulo and to rely on narrowing to provide both induction variables and instantiation schemas. Whenever the equational rewrite system $(\mathcal{R}, E)$ has good properties of termination, sufficient completeness, and when E is constructor and variable preserving, narrowing at defined-innermost positions leads to consider only unifiers which are constructor substitutions. This is especially interesting for associative and associative-commutative theories for which the general proof search system is refined. The method is shown to be sound and refutationaly correct and complete. A major feature of our approach is to provide a constructive proof in deduction modulo for each successful instance of the proof search procedure.

In [31] we describe a presentation of sequents in a two-dimensional space as well as a presentation of proofnets and sequent calculus derivations in a three-dimensional space. These renderings admit interesting geometrical properties: sequent occurrences appear as parallel segments in the case of three-dimensional sequent calculus derivations and the De Morgan duality is expressed by the fact that negation stands for a ninety degree rotation in the case of two-dimensional sequents and three-dimensional proofnets.

6.2. Integration of formal methods in programming languages

6.2.1. Formal islands and Tom

Participants: Emilie Balland, Paul Brauner, Horatiu Cirstea, Yves Guiraud, 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 [64]. The *Tom* system is documented in [28]. A general overview of the research problem raised by this approach are presented in [63].

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. During the internship of Nicolas Henry, we have developed a mapping to connect data-strucre generated by the Eclipse Modeling Framework to the *Tom* framework. The long-term goal of this project is to provide constructs to describe transformations of models, ion an expressive and safe way.

Tom is also a natural choice for querying and transforming structured data and in particular XML documents. In [23] we presented an extension of *Tom* that makes available all the *Tom* features into a syntax adapted to XML document manipulation. This extension can be in fact easily adapted to accommodate any other structured data provided that a tree representation can be obtained out of it.

We are currently working on the definition of a new type system for *Tom* along with the associated type inference and checking algorithms [27]. 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, Horatiu Cirstea, Claude Kirchner, 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 [10] and [38], [37]. 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 [14] we present the notion of anti-terms, i.e. terms that may contain complement symbols. We present algorithms for solving anti-pattern matching problems in the syntactic case as well as 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. 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, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau.

Security policies are one of the most fundamental elements of computer security. The rewrite-based approach of security policies provides executable specifications which can be independently designed, verified, and then anchored on programs using a modular discipline. In the lineage of [69], we describe in [16] 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 strategic 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.

Access control policies, a particular case of security policies should guarantee that information can be accessed only by authorized users and thus prevent all information leakage. We proposed [24], [21] 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.

In [26], extended Petri net processes are used to specify and verify security policies in a modular way. It defines fundamental policy properties, i.e., completeness, termination, consistency and confluence, in Petri net terminology and gets some theoretical results. According to XACML combiners and property-preserving Petri net process algebra, several policy composition operators are specified and property-preserving results are stated for the policy correctness verification. In [25], four types of policy compositions are defined, such that the integrated policy is capable of handling resources sharing, simultaneously executing operations and embedding sub-policies into main policies in multiple heterogeneous systems. Furthermore, the global policy can preserve the fundamental policy properties, (completeness, termination, consistency and confluence), and satisfy policy autonomy and security principles that are required for secure interoperation. These results are detailed and expanded in [32].

In a multiple domains application environment, where distributed multiple heterogeneous systems interoperate with each other, the local access control policies should correspondingly be integrated together in order to allow users of one organization to interact with other domains. One of the key challenges of integrating policies is the conflict detection and resolution while preserving individual policy consistency. The problem of detecting and resolving inheritance violation in the interoperation of multiple heterogeneous systems is addressed in [52]. The inheritance hierarchy of a security policy is formulated with a directed graph. Solving inheritance violation problem (IVP) is formulated as a feedback arc set problem, which is NP-hard. Then, some classical approximation algorithms are introduced. The IVP in two interoperating domains is

converted into the problem of finding a minimum weight vertex cover problem in a bipartite graph, which is polynomial-time solvable. These results are extended in [33] where several types of potential conflicts and consistency properties are considered. Graph theory, network flow technology and colored Petri nets are applied for specifying and verifying a secure interoperation design. The component-based integration of policies is applicable for both static and dynamic multi-domains environments.

The verification approaches presented above generally specify the analysed systems without making a clear distinction between the policies and the corresponding contexts. In [29] we propose a framework where the security policies and the systems they are applied on are specified separately but using a common formalism. This separation allows not only some analysis of the policy independently of the target system but also the application of a given policy on different systems. In this framework, we propose a method to check properties like confidentiality, integrity or confinment over secure systems based on different policy specifications.

7. Other Grants and Activities

7.1. Regional Initiatives

We obtained a financial support from the Lorraine region for funding the research activities of Tony Bourdier.

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. ANR Complice (2009-2012)

Participant: Yves Guiraud.

The ANR project "Complexité implicite, concurrence et extraction" (Complice), headed by Patrick Baillot (CNRS, LIP Lyon), federates researchers from Lyon (LIP), Nancy (LORIA) and Villetaneuse (LCR). The coordinator for the LORIA site is Guillaume Bonfante (Carte). The project held a meeting in Nancy on October 23.

7.2.2. 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.

7.2.3. ANR Ravaj (2007-2009)

Participants: Emilie Balland, 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.

7.2.4. ANR SSURF (2007-2009)

Participants: Tony Bourdier, Horatiu Cirstea.

"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.

7.2.5. ARC CORiAS(2009-2010)

Participants: Paul Brauner, Guillaume Burel, Horatiu Cirstea, Clément Houtmann, Claude Kirchner, Cody Roux.

The INRIA ARC "Conception et réalisation d'assistants à la preuve fondés sur la surdéduction modulo" (CORiAS), headed by Germain Faure (INRIA, Saclay), federates researchers from Nancy (LORIA) and Saclay (LIX). The coordinator for the LORIA site is Horatiu Cirstea. A detailed presentation is available at http://www.lix.polytechnique.fr/corias/.

7.2.6. ARC REDO (2009-2010)

Participant: Yves Guiraud.

The INRIA ARC "Redesigning logical syntax" (REDO), headed by Lutz Straßburger (INRIA, LIX Saclay), federates researchers from Bath (Department of Computer Science), Nancy (LORIA) and Saclay (LIX). The coordinator for the LORIA site is François Lamarche (Calligramme). The ARC held a meeting in Nancy on November 16-18.

7.2.7. FRAE QUARTEFT (2009-2012)

Participants: Jean-Christophe Bach, Horatiu Cirstea, Pierre-Etienne Moreau.

"QUARTEFT: QUAlifiable Real TimE Fiacre Transformations" is a research project founder by the FRAE (Fondation de Recherche pour l'Aéronautique et l'Espace). A first goal is to develop an extension of the Fiacre intermediate language to support real-time constructs. A second goal is to develop new model transformation techniques to translate this extended language, Fiacre-RT, into core Fiacre. A main difficulty consists in proposing transformation techniques that could be verified in a formal way.

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.

8. Dissemination

8.1. Animation of the scientific community

Guillaume Burel:

- Committee of Nancy INRIA Research Center until April 2009.

Horatiu Cirstea:

- Program committees of RuleML 2009 (International RuleML Symposium on Rule Interchange and Applications), RULE 2009 (International Workshop on Rule-Based Programming), WRS 2009 (Workshop on Reduction Strategies in Rewriting and Programming).
- Steering committee of RULE.

Yves Guiraud:

- Committee of Nancy INRIA Research Center.
- "Sustainable development" working group of INRIA Nancy.

Claude Kirchner:

- Director of the INRIA Bordeaux - Sud-Ouest research center.

Hélène Kirchner:

- Deputy scientific director at INRIA.

Pierre-Etienne Moreau:

- Program committee of SLE 2009 (International Conference on Software Language Engineering), CC 2008 (International Conference on Compiler Construction), LDTA 2009 (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.

Horatiu Cirstea:

- Master course in Nancy on programming and proving with rule based languages, with Pierre-Etienne Moreau.
- Supervision of Cyrille Cornu (internship École des Mines de Nancy).
- Supervision of Aymeric Carraro and Kévin Panier (internships Université Henri Poincaré)
- Supervision of Luc Sarzyniec and Cedric Lejault (internships ESIAL).

Yves Guiraud:

- Supervision of Sébastien Martinelle and David Serra (internship ESIAL Nancy).

Pierre-Etienne Moreau:

- Master course in Nancy on programming and proving with rule based languages, with Horatiu Cirstea,
- Supervision of Nicolas Henry's internship (Université Henri Poincaré)
- Supervision of Cynthia Florentin's internship (Université Henri Poincaré)

Clément Houtmann:

- Supervision of Thomas Boudin (internship École des Mines de Nancy)

8.3. Communications in conferences

Horatiu Cirstea:

- "TomML: A Rule Language For Structured Data", International RuleML Symposium on Rule Interchange and Applications, Las Vegas, US, November 5-7, 2009.

Yves Guiraud:

- "Complexity of polygraphic programs", ANR Complice meeting, Université Paris 13, Villetaneuse, January 27.
- "Higher-dimensional categories with finite derivation type", Logic Seminar, Institut de Mathématiques de Luminy, Université Aix-Marseille 2, April 9.
- "Finite derivation type and identities among relations for higher-dimensional rewriting systems", Workshop on Computer Algebra Methods and Commutativity of Algebraic Diagrams (CAM-CAD), Institut de Recherche en Informatique de Toulouse, Université Toulouse 3, October 16-17.

Pierre-Etienne Moreau:

- “Introduction aux langages dédiés”, GDR Robotique.

8.4. Visits

Yves Guiraud:

- Institut Camille Jordan, Lyon: June 8-12.

Claude Kirchner

- SRI International, one week in August.

8.5. Visitors

- Philippe Malbos, Institut Camille Jordan, Université Lyon 1: May 13-20 and October 5-9.
- Alberto Naibo, Université Paris 1 Sorbonne: July 9-10.

8.6. Theses

- Guillaume Burel: “Bonnes démonstrations en déduction modulo”, PhD
- Emilie Balland: “Conception d’un langage dédié à l’analyse et la transformation de programmes”, PhD

8.7. Thesis and admission committees

Horatiu Cirstea:

- Member of the recruitment committee for an Associate Professor position at ENSEM (INPL).

Claude Kirchner:

- Emilie Balland: “Conception d’un langage dédié à l’analyse et la transformation de programmes”, PhD (co-advisor)
- Guillaume Burel: “Bonnes démonstrations en déduction modulo”, PhD (advisor)

Pierre-Etienne Moreau:

- Emilie Balland: “Conception d’un langage dédié à l’analyse et la transformation de programmes”, PhD (co-advisor)
- Muck Joost van Weerdenburg: “Implementation of Rewrite Systems”, PhD

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, jul 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", vol. 5, n^{o} 2:14, 2009, p. 1-37.

[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, http://dx.doi.org/10.1109/LICS.2007.37.

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

[7] Y. GUIRAUD, P. MALBOS. *Higher-dimensional categories with finite derivation type*, in "Theory and Applications of Categories", vol. 22, n^{o} 18, 2009, p. 420-478.

[8] 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.

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

Year Publications

Doctoral Dissertations and Habilitation Theses

[10] E. BALLAND. *Conception d’un langage dédié à l’analyse et la transformation de programmes*, Université Henri Poincaré - Nancy I, 03 2009, http://tel.archives-ouvertes.fr/tel-00435881/en/, Ph. D. Thesis.

[11] G. BUREL. *Bonnes démonstrations en déduction modulo*, Université Henri Poincaré - Nancy I, 03 2009, http://tel.archives-ouvertes.fr/tel-00372596/en/, Ph. D. Thesis.

Articles in International Peer-Reviewed Journal

[12] G. BONFANTE, Y. GUIRAUD. *Polygraphic programs and polynomial-time functions*, in "Logical Methods in Computer Science (LMCS)", vol. 5, n^{o} 2:14, 2009, p. 1-37, http://hal.inria.fr/inria-00122932/en/.

[13] G. BUREL, C. KIRCHNER. *Regaining Cut Admissibility in Deduction Modulo using Abstract Completion*, in "Information and Computation", 2009, In Press, http://hal.inria.fr/inria-00132964/en/.

[14] H. CIRSTEA, C. KIRCHNER, R. KOPETZ, P.-E. MOREAU. *Anti-patterns for Rule-based Languages*, in "Journal of Symbolic Computation", 2009, http://hal.inria.fr/inria-00429226/en/.

[15] Y. GUIRAUD, P. MALBOS. *Higher-dimensional categories with finite derivation type*, in "Theory and Applications of Categories", vol. 22, n^{o} 18, 2009, p. 420-478, http://hal.archives-ouvertes.fr/hal-00326974/en/.

[16] C. KIRCHNER, H. KIRCHNER, A. SANTANA DE OLIVEIRA. *Analysis of Rewrite-Based Access Control Policies*, in "Electronic Notes in Theoretical Computer Science", vol. 234, 2009, p. 55-75, http://hal.inria.fr/inria-00433409/en/.

[17] F. NAHON, C. KIRCHNER, H. KIRCHNER, P. BRAUNER. *Inductive Proof Search Modulo*, in "Annals of Mathematics and Artificial Intelligence", vol. 55, n^{o} 1, 2009, p. 123-154, http://hal.inria.fr/inria-00337380/en/.

International Peer-Reviewed Conference/Proceedings

[18] O. ANDREI, H. KIRCHNER. *A Port Graph Calculus for Autonomic Computing and Invariant Verification*, in "TERMGRAPH 2009, 5th International Workshop on Computing with Terms and Graphs, Satellite Event of ETAPS 2009", To appear in Electronic Notes in Theoretical Computer Science, Elsevier, 2009, http://hal.inria.fr/inria-00418560/en/.

[19] F. BLANQUI, C. ROUX. *On the relation between sized-types based termination and semantic labelling*, in "18th EACSL Annual Conference on Computer Science Logic - CSL 09, Portugal Coimbra", 2009, http://hal.inria.fr/inria-00397689/en/CN.

[20] T. BOURDIER, H. CIRSTEA, D. J. DOUGHERTY, H. KIRCHNER. *Extensional and Intensional Strategies*, in "9th International Workshop on Reduction Strategies in Rewriting and Programming - WRS'09, Brésil Brasilia", 2009, http://hal.inria.fr/inria-00429235/en/.

[21] T. BOURDIER, H. CIRSTEA, P.-E. MOREAU, A. SANTANA DE OLIVEIRA. *Analysis of Lattice-Based Access Control Policies using Rewiting Systems and Tom.*, in "1st Luxembourg Day on Security and Reliability, Luxembourg Luxembourg city", 2009, http://hal.inria.fr/inria-00433424/en/.

[22] G. BUREL. *Automating Theories in Intuitionistic Logic*, in "7th International Symposium on Frontiers of Combining Systems -FroCoS'09, Italie Trento", S. GHILARDI, R. SEBASTIANI (editors), vol. 5749, Springer, 2009, p. 181-197, http://hal.inria.fr/inria-00395934/en/.

[23] H. CIRSTEA, P.-E. MOREAU, A. REILLES. *TomML: A Rule Language For Structured Data*, in "International RuleML Symposium on Rule Interchange and Applications - RuleML 2009, États-Unis d'Amérique Las Vegas", 2009, p. 262-271, http://hal.inria.fr/inria-00429229/en/.

[24] H. CIRSTEA, P.-E. MOREAU, A. SANTANA DE OLIVEIRA. *Rewrite Based Specification of Access Control Policies*, in "3rd International Workshop on Security and Rewriting Techniques - SecReT 2008, États-Unis d'Amérique Pittsburgh", vol. 234, 2009, p. 37-54, http://hal.inria.fr/inria-00335091/en/.

[25] H. HUANG, H. KIRCHNER. *Policy Composition based on Petri Nets*, in "33rd Annual IEEE International Computer Software and Applications Conference COMPSAC2009, États-Unis d'Amérique Seattle", IEEE Computer Society Press, 2009, p. 416–421, http://hal.inria.fr/inria-00433398/en/.

[26] H. KIRCHNER, H. HUANG. *Component-based Security Policy Design with Colored Petri Nets*, in "Semantics and Algebraic Specification, Italie Udine", J. PALSBBERG (editor), vol. 5700, Springer, Marina Lenisa, 2009, p. 21-42, http://hal.inria.fr/inria-00433372/en/.

[27] C. KIRCHNER, P.-E. MOREAU, C. TAVARES. *A Type System for Tom*, in "The Tenth International Workshop on Rule-Based Programming, Brésil Brasilia", 2009, http://hal.inria.fr/inria-00426439/en/.

Research Reports

[28] J.-C. BACH, E. BALLAND, P. BRAUNER, R. KOPETZ, P.-E. MOREAU, A. REILLES. *Tom Manual*, 2009, http://hal.inria.fr/inria-00121885/en/, Rapport Technique.

Other Publications

[29] T. BOURDIER, H. CIRSTEA, M. JAUME, H. KIRCHNER. *On Formal Specification and Analysis of Security Policies*, 2009, http://hal.inria.fr/inria-00429240/en/.

[30] Y. GUIRAUD, P. MALBOS. *Identities among relations for higher-dimensional rewriting systems*, 2009, http://hal.archives-ouvertes.fr/hal-00426228/en/, Preprint, 16 pages.

[31] C. HOUTMANN. *Three Dimensional Proofnets for Classical Logic*, 2009, http://hal.inria.fr/inria-00426469/en/.

[32] H. HUANG, H. KIRCHNER. *Modular Security Policy Design based on Extended Petri Nets*, 2009, http://hal.inria.fr/inria-00396924/en/.

[33] H. HUANG, H. KIRCHNER. *Secure Interoperation in Heterogeneous Systems based on Colored Petri Nets*, 2009, http://hal.inria.fr/inria-00396952/en/.

References in notes

[34] *The Coq Proof Assistant, version 8.2*, 2009, http://coq.inria.fr.

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

[36] 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, Haifa Israël", 2008, http://hal.inria.fr/inria-00328554/en/.

[37] 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/.

[38] 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/.

[39] 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/.

[40] G. BONFANTE, Y. GUIRAUD. *Intensional properties of polygraphs*, in "Electronic Notes in Theoretical Computer Science", vol. 203, n^{o} 1, 2008, p. 65-77.

[41] 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, sep 1996.

[42] 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, Jul 2002, p. 155-185.

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

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

[45] 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.

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

[47] 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, vol. 308, Springer-Verlag, jul 1987, p. 258–263, Also as internal report CRIN: 88-R-001.

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

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

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

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

[52] H. HUANG, H. KIRCHNER, S. LIU, W. WU. *Handling Inheritance Violation for Secure Interoperation of Heterogeneous Systems*, in "International Journal of Security and Networks", vol. 4, 03 2007, p. 223-233, http://hal.inria.fr/inria-00433391/en/, D.: Software This work was supported in part by National Natural Science Foundation of China under Grant 10701030, INRIA, and National Science Foundation of USA under grants CNS-0524429 and CCF-0627233..

[53] 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.

[54] 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.

[55] 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.

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

[57] 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.

[58] 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), sep 1993.

[59] 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.

[60] 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, http://dx.doi.org/10.1016/j.tcs.2008.01.019.

[61] 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.

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

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

[64] 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/.

[65] 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.

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

[67] M. PARIGOT. *Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction*, in "Logic Programming and Automated Reasoning, St Petersburg, Russia", A. VORONKOV (editor), Springer, 1992, p. 190-201.

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

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

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

[71] 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.