INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Contraintes

Constraint Programming

Rocquencourt

THEME 2A

Activity Report

2003

Table of contents

1. Team

Head of project-team
François Fages [DR, INRIA]

Vice-head of project-team
Pierre Deransart [DR, INRIA]

Administrative assistant
Josy Baron [up to 31/1/3]
Nadia Mesrar [from 1/4/3 to 15/7/3]
Emmanuelle Grousset [from 6/10/3]

Staff member
Sylvain Soliman [CR, INRIA]

Project technical staff
Guillaume Arnaud [IE, INRIA]

Collaborators
Frédéric Benhamou [Professor, University of Nantes]
Philippe Codognet [Professor, University of Paris 6, up to 30/6/3]
Daniel Diaz [Associate Professor, University of Paris 1]
Gérard Ferrand [Professor, University of Orléans]

Ph.D. students
Nathalie Chabrier-Rivier [INRIA scholarship]
Emmanuel Coquery [MESR scholarship up to 30/9/3, ATER CNAM since]
Sorin Craciunescu [Ecole Polytechnique, ATER University of Paris 12]
Rémy Haemmerlé [INRIA scholarship, since 1/11/3]
Ludovic Langevine [INRIA scholarship]

Student interns
Annabelle Ballesta [INSA Rouen, from 7/7/3 to 31/8/3]
Rémi Coolen [Ecole Polytechnique, from 14/4/3 to 15/7/3]
Daniela Grosu [DEA OJMA, from 1/5/3 to 30/9/3]
Rémy Haemmerlé [DEA SPL, from 7/4/3 to 30/9/3]
Sumit Kumar [IIT Kanpur, from 12/5/3 to 31/7/3]
Matthieu Pierrès [DEA IARFA, from 1/4/3 to 31/12/3, shared with Fractale/Complex team]
Nathalie Sznajder [MMFAI ENS Paris, from 10/6/3 to 16/9/3]

2. Overall Objectives

The "Contraintes" group investigates the logical foundations, design, implementation, programming environments and applications of constraint programming languages.

Constraint Programming is a field born during the mid 80s from Logic Programming, Linear Programming coming from Operations Research, and Constraint Propagation techniques coming from Artificial Intelligence. Its foundation is the use of relations on mathematical variables to compute with partial information.

The successes of Constraint Programming in industrial applications are related to the bringing of new local consistency techniques and of *declarative languages* which allow an easier control on the mixing of heterogeneous resolution techniques: numerical, symbolic, deductive, heuristic, etc.

The study of Concurrent Constraint languages is a core aspect of the project as they provide a conceptual framework for analyzing different issues of Constraint Programming, like constraint resolution techniques, concurrent modeling, reactive applications, etc. The main application domains investigated are combinatorial optimization and bioinformatics.

3. Scientific Foundations

3.1. Concurrent constraint programming

The class of Concurrent Constraint programming languages (CC) was introduced a decade ago by Vijay Saraswat as a unifying framework for constraint logic programming and concurrent logic programming. The CC paradigm constitutes a representative abstraction of constraint programming languages, and thus allows a fine grained study of their fundamental properties.

CC generalizes the Constraint Logic Programming framework (CLP) by introducing a synchronization primitive, based on constraint entailment. It is a model of concurrent computation, where agents communicate through a shared store, represented by a constraint, which expresses some *partial information* on the values of the variables involved in the computation. The variables play the role of transmissible dynamically created communication channels.

One of the big successes of CC has been the simple and elegant reconstruction of finite domain constraint solvers, and the cooperation of several models to solve a single combinatorial problem. On the other hand, to use CC for programming reactive applications forces one to abandon the hypothesis of monotonic evolution of the constraint store; this is a strong motivation for new extensions of CC languages.

There are strong completeness theorems relating the execution of a CLP program and its translation in classical logic, which provide smooth reasoning techniques for such programs. However these theorems are broken by the synchronization operation of CC. Looking for a logical semantics of CC programs in the general paradigm of logic programming, *program = formula, execution = proof search*, leads to a translation in Jean-Yves Girard's linear logic. This allows the recovery of some completeness results about successes and stores; even suspensions may be characterized with the non-commutative logic of Ruet and Abrusci.

It is thus possible to address important issues for Constraint Programming:

- verifying CC programs;
- combining CLP and state-based programming;
- dealing with local search inside a global constraint solving procedure.

The last two cases rely on a natural extension of CC languages, called Linear Concurrent Constraint languages (LCC), which simply replaces constraint systems built onto classical logic by constraint systems built onto linear logic. This allows us to represent state changes thanks to the consumption of resources during the synchronization action, modeled by the linear implication.

3.2. Constraint solvers

Our domains of application use quite different constraint systems:

- finite domains (bounded natural numbers): primitive constraint of some finite domain membership, numerical, symbolic, higher order and global constraints;
- reals: Simplex algorithm for linear constraints and interval methods otherwise;
- terms: subtyping constraints and ontologies.

The project works on both the constraint resolution methods and their cooperation. The main focus is the efficiency of constraint propagation methods, and their combination with local search methods.

3.3. Constraint programming environments

Handling hundreds of thousands of heterogeneous constraints on as many variables is impossible without tools specially designed to show the different views of the execution.

The DiSCiPl ESPRIT project showed the use of declarative diagnosis methods based on the logical semantics of programs for interactive debugging systems, and developed tools to visualize the execution of CLP programs: showing the effect of propagation, the shape of the search space (detection of symmetries), the impact of heuristics, etc. We have pursued this research with our partners of the RNTL project OADymPPaC. The main results of this project are a generic trace format for constraint programming languages and a large investigation of visualization tools.

Semantics based debugging, static and dynamic typing, and more generally validation of CC programs are also studied in the project. Concerning typing issues in constraint programming, the usual coercions between constraint domains (e.g. between booleans and integers, lists and terms) offer a particularly challenging task to type systems design.

4. Application Domains

4.1. Combinatorial optimization

The number and economic impact of combinatorial optimization problems found in the industrial world are constantly increasing. They cover:

- resource allocation;
- placement;
- scheduling;
- parallelization;
- transport;
- etc.

The last forty years have brought many improvements in Operations Research resolution techniques. In this framework, Constraint Programming can be seen as providing, on the one hand, local coherence techniques that can be applied to various numerical or symbolic constraints, and on the other hand, declarative languages. This last point is crucial for quickly developing complex combinations of algorithms, which is not possible without a language with a high level of abstraction. It allowed for better results, for instance in scheduling problems, than traditional methods, and is promised to an even better future when thinking about cooperation of global resolution and local propagation techniques.

The project builds upon its knowledge of CC languages, constraint solvers and their implementation to work in these directions. The LCC languages offer a framework for theoretical analysis; and the soft constraints framework gives a new way to tackle over-constrained problems. The work on programming environments helps to integrate the Constraint Programming tools into this application domain.

4.2. Bioinformatics

In recent years, Biology began a work of elucidation of high level biological processes in terms of their biochemical bases at the molecular scale. At the end of the Nineties, research in Bioinformatics evolved, passing from the analysis of the genomic sequence to the analysis of various data produced in mass by *post-genomic* technologies (expression of ARN and proteins, SNP and haplotypes, protein-protein interactions, 3D structures, etc). The complexity of the systems concerned requires a large research effort to develop the symbolic notation of biological processes and data.

In 2002 , we have started a Collaborative Research Initiative ARC CPBIO on "Process Calculi and Biology of Molecular Networks". By working with biologists on well understood biological models, we seek:

- to identify in the family of competitive models coming from the Theory of Concurrency (Pi-calculus, Join-calculus and their derivatives) and from Logic Programming (Constraint Logic Programming, Concurrent Constraint languages and their extensions to discrete and continuous time, TCC, HCC), the ingredients of a language for the modular and multi-scale representation of biological processes;
- to provide a series of examples of biomolecular processes transcribed in formal languages, and a set of biological questions of interest about these models;
- to design and apply to these examples formal computational reasoning tools for the simulation, the analysis and the querying of the models.

5. Software

5.1. GNU-Prolog

Participants: Daniel Diaz, Rémy Haemmerlé.

GNU Prolog is a free Prolog compiler with constraint solving over finite domains developed by Daniel Diaz. GNU Prolog accepts Prolog extended with primitives for constraint programming and produces native binaries (like gcc does from a C source). The Prolog part conforms to the ISO standard for Prolog with many practical extensions (global variables, OS interface, sockets,...). GNU Prolog also includes an efficient constraint solver over Finite Domains (FD), giving the user the combined power of constraint programming and the declarativity of logic programming.

An experimental version called GNU-Prolog-RH has also been released and distributed this year [25]. This version developed by Rémy Haemmerlé is an extension of GNU-Prolog with attributed variables, coroutining and constraint logic programming over reals. This version greatly extends the expressive and modeling power of GNU-Prolog. It is also used to prototype the design and implementation of our new Silcc language.

5.2. BIOCHAM

Participants: Nathalie Chabrier-Rivier, François Fages, Sylvain Soliman.

The Biochemical Abstract Machine BIOCHAM is a programming environment for modeling biochemical systems, making simulations and querying the model in temporal logic CTL.

In its initial version BIOCHAM 0.1 is composed of:

- a simple rule-based language for modeling biochemical systems at a boolean abstraction level,
- a simple simulator,
- a powerful query language based on temporal logic CTL for querying the temporal properties of the model,
- an interface to the NuSMV model checker for evaluating CTL queries.

An experimental interface to LOTOS has been developed in collaboration with the VASY project-team. A repository of computational models of biological systems called CMBSlib aimed at comparing models as well as formalisms, is currently under construction in collaboration with our partners of the ARC CPBIO.

5.3. Adaptive search library

Participants: Daniel Diaz, Philippe Codognet.

This is a C library to solve constraint satisfaction problems by the adaptive local search method. The current release is limited to permutation problems. More precisely, all n variables have the same domain $x_1..x_n$ and are subject to an implicit all-different constraint. Several problems fall into this category and some examples are provided with the library.

5.4. TCLP

Participant: Emmanuel Coquery.

TCLP is a prescriptive type system for Constraint Logic Programming, currently: ISO-Prolog, GNU-Prolog, SICStus Prolog and the constraint programming libraries of SICStus Prolog. The flexibility of type checking in TCLP is due to three kinds of polymorphism: parametric polymorphism (e.g. *list(A)*), subtyping (e.g. *list(A)<term*) and overloading (e.g. $-$*:num*$*$*num*$\rightarrow$*num* and $-$*:A*$*$*B*$\rightarrow$*pair(A,B)*). No type declaration are required, thanks to the type inference algorithm for predicates and to a default *term* type for function symbols.

5.5. CLPGUI

Participant: François Fages.

CLPGUI is a generic graphical user interface written in Java for constraint logic programming. It is currently available for GNU-Prolog and Sicstus Prolog. CLPGUI has been developed both for teaching purposes and for debugging complex programs. The graphical user interface is composed of several windows: one main console and several dynamic 2D and 3D viewers of the search tree and of finite domain variables. With CLPGUI it is possible to execute incrementally any goal, backtrack or recompute any state represented as a node in the search tree. The level of granularity of the search tree is defined by annotations in the CLP program.

6. New Results

6.1. Linear Concurrent Constraint programming

Participants: Emmanuel Coquery, François Fages, Rémy Haemmerlé, Sylvain Soliman.

Now that the theory of LCC languages in Linear Logic is well established, we have started the development of a first implementation of this paradigm. The system named Silcc (pronounce "silk", meaning "Silcc Is LCC") is a complete implementation of LCC based on GNU-Prolog native code compiling technology. Silcc is an extensible and modular constraint programming system which is based on a small kernel language, called LCC(K) [25]. Several libraries will be developed on top of this kernel for: constraint solving over different domains, search procedures, forward chaining rules and compatibility with other systems like GNU-Prolog. The main novelties for the programmer will include:

- imperative features unified with constraints,
- powerful synchronization primitives,
- capability of defining new constraint systems,
- accompanying verification tools.

6.2. Contextual Logic Programming

Participant: Daniel Diaz.

While a program-structuring feature is required for a production programming language, the current proposals for the inclusion of modules in the ISO Prolog standard are not very consensual. We have thus investigated an alternative solution based on Contextual Logic Programming (CxLP). Informally, the main point of CxLP is that programs are structured as sets of predicates (*units*) which can be dynamically combined in an execution attribute called a *context*. Goals are seen just as in regular Prolog, except for the fact that the matching predicates are to be located in all the units which make up the current context. We extended CxLP to attach *arguments* to units: these serve the dual purpose of acting as "unit-global" variables and as state placeholders in actual contexts. CxLP clearly carries a higher overhead than regular Prolog, as the context must be searched at run-time for the unit that defines a goal's predicate, a process which requires at least one extra indirection compared to straight Prolog; this kind of situation has become more usual and less of a performance issue in recent systems, in Object-Oriented and even in procedural languages, for instance as a result of using dynamically-loaded shared libraries. We have built a prototype implementation of a Contextual Logic Programming language inside GNU-Prolog [2], which has surprisingly good performance, considering we aren't performing any optimization.

6.3. Subtyping constraints in quasi-lattices

Participants: Emmanuel Coquery, François Fages.

We have shown the decidability and NP-completeness of the satisfiability problem for non-structural subtyping constraints in quasi-lattices [10]. This problem, first introduced by Smolka in 1989, is important for the typing of logic and functional languages. We have generalized Trifonov and Smith's algorithm over lattices, to the case of quasi-lattices with a complexity in $O(m^v M^v n^3)$, where m (resp. M) stands for the number of minimal (resp. maximal) elements of the quasi-lattice, v is the number of unbounded variables and n is the number of constraints. Similarly, we have extended Pottier's algorithm for computing explicit solutions to the case of quasi-lattices.

These results directly apply to our system TCLP for type inferencing in constraint logic programming [8]. Their application to ontological reasoning is under investigation.

6.4. The cutset global constraint

Participant: François Fages.

We have designed a new global constraint for cutset problems. A cutset in a directed graph $G = (V, E)$ is a set of vertices that cuts all cycles in G. Finding a cutset of minimum cardinality is NP-hard. There exist several approximate algorithms and exact algorithms, most of them using graph reduction techniques. The cutset constraint we propose in [16] is a boolean constraint over variables associated to the vertices of a given graph, that states that the subgraph restricted to the vertices having their boolean variable set to true is acyclic. We propose a filtering algorithm based on graph contraction operations and inference of simple boolean constraints, that has a linear time complexity in $O(|E| + |V|)$. Several search heuristics based on graph properties provided by the cutset constraint have been studied. The efficiency of the cutset constraint is shown on on benchmarks of the literature for pure minimum cutset problems, and on an application to log-based reconciliation problems where the global cutset constraint is mixed with other boolean constraints.

6.5. Adaptive search

Participants: Philippe Codognet, Daniel Diaz, François Fages, Matthieu Pierrès.

We have re-implemented the Adaptive Search method as a C-based framework library [7]. This new implementation is more generic and efficient than the previous one but is dedicated to permutation problems, that is: all variables have a same initial domain and are subject to an implicit all-different constraint. Many classical problems fall into this category. In this new implementation, stochastic moves can also occur to escape from a plateau with a given probability. This last action has proved to be very effective on benchmarks where many local minima with large plateaus occur such as the magic square problem for which a ten time speedup factor can be achieved.

This adaptive search method has been combined with genetic algorithms in a multi-user system, developed in collaboration with the Fractale/Complex team, for assigning offices to project-teams at INRIA Rocquencourt. In this experimental system, sets of solutions satisfying preference constraints are computed by adaptive search, and the preferences of users are learned by genetic algorithms through their notation of the solutions proposed by the system [26].

6.6. Detection of symmetries

Participants: Guillaume Arnaud, Annabelle Ballesta, Pierre Deransart, François Fages.

Symmetries in constraint satisfaction problems are an important issue. They may lead to a lot of redundant computations and dramatically increase the size of the search space (for example simple instances of the Balanced Incomplete Block Design problem have more than 10^{52} symmetries). The work on symmetries is twofold: first, the symmetries of a specific problem have to be identified, then they have to be broken to avoid useless computation. We have developed a library for GNU-Prolog that implements the classical SBDS algorithm (Symmetry Breaking During Search). This library provides an easy symmetry breaking facility for GNU-Prolog programmers [3]. Besides, a first algorithm for statically detecting symmetries in Constraint Satisfaction Problems has been investigated in [20].

6.7. Generic trace format

Participants: Pierre Deransart, François Fages, Gérard Ferrand, Ludovic Langevine.

Debugging tools are essential to help tune constraint solving programs and each platform has its own environment tools. However, at present, these tools are specific and have to be redesigned and re-implemented for each constraint solver whereas much could be factorized. We propose a generic framework to enable debugging tools to be defined almost independently from finite domain solvers, and conversely, tracers to be built independently from these tools. We have designed a *generic trace schema* based on a *generic observational semantics* which formalizes relevant aspects of constraint programming and solving. We have developed Codeine, a tracer for the GNU-Prolog platform, that implements this schema [18][19]. The genericity of the schema has also been demonstrated on two other platforms (Choco and PaLM) by our partners of the OADymPPaC Project.

6.8. Visualization for debugging

Participants: Guillaume Arnaud, Pierre Deransart, François Fages, Ludovic Langevine.

Using generic trace format allows us to develop solvers and debugging tools independently. It is a way to incentivate the creation of new and more powerful debugging tools. To validate this approach we experimented several combinations of new debugging tools (from the OADymPPaC project partners) with some solvers, especially GNU-Prolog and PaLM which handle constraint propagation in different ways [12]. We have shown that they all find in the generic trace the whole information they require. The debugging tools are mostly visualization oriented: they display abstractions of the execution. The developer can use these displays to better understand the behavior of his program and to correct it or to improve its performances. Two new viewers have been implemented by us to display search-trees. The first viewer is an adaptation of the Christmas-Tree developed by ILOG, using propagation details to assign a more or less big size to each node. The second viewer, River-Tree, applies the so-called Strahler numbers as a clue to guide in the exploration of the search-tree and get a better overall impression of it. Strahler numbers were originally used to relate the morphological structure of river networks: they give quantitative measures of the complexity of a tree and its sub-trees. The integration of these viewers in a new released version of CLPGUI is still in progress.

6.9. 3D placement constraints and virtual reality

Participants: Rémi Coolen, François Fages.

We have investigated a constraint programming approach to a 3D placement problem (placing hardware in an office) combined with virtual reality tools for the visualization of the problem and complex interactions. The idea is to explore the concept of constraint programming as a paradigm of augmented virtual reality. The user interacts with the placement system through a representation of the scene in virtual reality, augmented with a representation of distance constraints [22].

6.10. Symbolic model checking of biomolecular interaction nets

Participants: Nathalie Chabrier-Rivier, François Fages, Daniela Grosu, Nathalie Sznajder, Sylvain Soliman.

In [1] we have introduced a formalism to represent and analyze protein-protein and protein-DNA interaction networks. The expressivity of this language, is illustrated by providing a formal counterpart of Kohn's compilation on the mammalian cell cycle control. This effectively turns an otherwise static knowledge into a discrete transition system incorporating a qualitative description of the dynamics. Our proposal is then to use the Computation Tree Logic CTL as a query language for querying the possible behaviors of the system. We provide examples of biologically relevant queries expressed in CTL about the mammalian cell cycle control and show the effectiveness of symbolic model checking tools to evaluate CTL queries in this context [5]. The generalization of this approach to quantitative models using differential equations [5][24] and to probabilistic models using probabilistic model checking [28] is under investigation.

The biochemical abstract machine BIOCHAM 0.2 is based on these concepts [6][21].

7. Contracts and Grants with Industry

7.1. COSYTEC

Collaboration within the RNTL project OADymPPaC (Nov. 2000 - May 2004)

Technology transfer of the generic trace format for the CHIP product.

Technology transfer of the CLPGUI software for the dynamic visualization of CHIP program execution.

7.2. ILOG

Collaboration within the RNTL pre-competitive project Manifico (Feb. 2003 - Feb. 2006, 150 Keuros) on non intrusive metacompilation of matching with constraints in rule based languages

Collaboration within the RNTL project OADymPPaC, evaluation of Ilog Discovery visualization component in the framework of constraint programming

8. Other Grants and Activities

8.1. National contracts

- RNTL project MANIFICO (Sep. 2003-2006) on the compilation of rules and constraints. with LORIA PROTHEO and ILOG coord.
- INRIA cooperative research initiative ARC CPBIO (2002-2004) on "Process Calculi and Molecular Networks Biology", with LORIA MODBIO, CNRS PPS lab. of University Paris 7 and Genoscope Evry, coord. F. Fages INRIA Rocquencourt.
- RNTL project OADymPPaC (Nov. 2000 - May 2004) "Tools for Dynamic Analysis and Debugging of Constraint Programs" with IRISA LANDES/INSA, INRIA-Futurs, Ecole des Mines de Nantes, University of Orléans, COSYTEC and ILOG, coord. P. Deransart INRIA Rocquencourt.

8.2. European contracts

- 6th PCRD STREP APRIL II “Applications of probabilistic inductive logic programming”, coord. Prof. L. de Raedt, University of Freiburg.
- 6th PCRD Network of Excellence REWERSE “Reasoning with rules and semantics”, coord. Prof. F. Bry, Ludwig Maximillian’s University in Munich.
- 5th PCRD Network of Excellence COLOGNET “Computational logic network”, area of constraint logic programming coord. F. Rossi, University of Padova.
- ERCIM working group on Constraints, coord. F. Fages, INRIA Rocquencourt.

8.3. Invitations

Prof. Stephen Muggleton from the Computational Bioinformatics Group at Imperial College in London, visited our group in August.

9. Dissemination

9.1. Teaching

Contraintes is affiliated to the Doctoral school EDITE of the University of Paris 6.

All Ph.D. students and some members of Contraintes teach in the first or second cycles of Universities or Engineering schools. Our involvement in third cycle cursus is the following:

- DEA SPL “Sémantique, Preuve et Langages”, ENS, X, University Paris 6, 7, 11: course of programmation par contraintes, François Fages 10h, Sylvain Soliman 10h.
- DEA IARFA, University Paris 6: Philippe Codognet 20h.
- DEA Informatique, University of Orléans: François Fages 3h.
- DESS Informatique, University Paris Sorbonne: Daniel Diaz 30h.
- Ecole Jeunes Chercheurs en Programmation, course of programmation par contraintes, François Fages 6h.

9.2. Leadership within scientific community

- Pierre Deransart is the General Secretary, past Chairman, of the “Association Française pour la Programmation en Logique et la programmation par Contraintes” AFPLC, and member of the Steering Committee of PPDP. He is in charge of international relationships of INRIA with Brazil and Portugal. He is the manager of the OADymPPaC project.
- François Fages is the Chairman of the ERCIM Working Group on Constraints, the Chairman of the “Association Française pour la Programmation en Logique et la programmation par Contraintes” AFPLC, a member past-Chairman of the Steering Committee of the ACM International Conference on Principles and Practice of Declarative Programming, PPDP, and a member of the Scientific Council of the French-Russian Liapunov Institute.
- Sylvain Soliman is the Secretary of the ERCIM Working Group on Constraints.

10. Bibliography

Articles in referred journals and book chapters

[1] N. Chabrier, M. Chiaverini, V. Danos, F. Fages, V. Schächter. *Modeling and querying biochemical networks.* in « Theoretical Computer Science », volume To appear, 2003.

Publications in Conferences and Workshops

[2] S. Abreu, D. Diaz. *Objective: in Minimum Context.* in « Proceedings of ICLP'2003, International Conference on Logic Programming », MIT Press, Mumbai, India, 2003.

[3] G. Arnaud. *Implantation de SBDS en GNU-Prolog.* in « Programmation en logique avec contraintes, proceedings of JFPLC 2003 », series special issue of RSTI, Hermès, M. Ducassé, editor, pages 237–250, Amiens (France), June, 2003.

[4] N. Chabrier, F. Fages. *"Model-checking" symbolique de réseaux biochimiques.* in « Actes des Journées Francophones de la Programmation en Logique avec Contraintes JFPLC'2003 », series special issue of RSTI, Hermès, M. Ducassé, editor, pages 155–168, Amiens (France), June, 2003.

[5] N. Chabrier, F. Fages. *Symbolic model cheking of biochemical networks.* in « Proceedings of the first Workshop on Computational Methods in Systems Biology, CMSB'03 », series LNCS, volume 2602, Springer-Verlag, pages 149–162, Rovereto, Italy, March, 2003.

[6] N. Chabrier, F. Fages. *The Biochemical Abstract Machine BIOCHAM.* in « Poster proceedings of European Conference on Computational Biology ECCB'03 », C. Christophe, H. Lenhof, M. Sagot, editors, Paris, September, 2003.

[7] P. Codognet, D. Diaz. *An Efficient Library for Solving CSP with Local Search.* in « Proc. Metaheuristics International Conference MIC'03 », Kyoto, Japan, 2003.

[8] E. Coquery. *TCLP: A type checker for CLP(X).* in « Proceedings of the the 13th workshop on logic programming environments, associated to ICLP'03 », A. Serebrenik, F. Mesnard, editors, Mumbai, India, December, 2003.

[9] E. Coquery, F. Fages. *Contraintes de sous-typage dans les quasi-treillis.* in « Actes des Journées Francophones de la Programmation en Logique avec Contraintes JFPLC'2003 », series special issue of RSTI, Hermès, M. Ducassé, editor, pages 253–266, Amiens (France), June, 2003.

[10] E. Coquery, F. Fages. *Subtyping constraints in quasi-lattices.* in « Proceedings of the 23rd conference on foundations of software technology and theoretical computer science, FSTTCS'2003 », series LNCS, Springer-Verlag, P. Pandya, J. Radhakrishnan, editors, Mumbai, India, December, 2003.

[11] R. Debruyne, G. Ferrand, N. Jussien, W. Lesaint, S. Ouis, A. Tessier. *Correctness of constraint retraction algorithms.* in « Proceedings of FLAIRS'03: 6th international Florida Artificial Intelligence Research Society conference », AAAI press, pages 172–176, 2003.

[12] P. DERANSART, L. LANGEVINE, M. DUCASSÉ. *Debugging Constraint Problem with Portable Tools.* in « Proceedings of the International Workshop on Logic Programming Environments WLPE'03 associated to ICLP'03 », Tata Institute of Fundamental Research, A. SEREBRENIK, editor, Mumbai, India, December, 2003.

[13] P. DERANSART, L. LANGEVINE, M. DUCASSÉ. *Generic Trace Model for Finite Domain Solvers and its Application to GNU-Prolog.* in « Proceedings of the Joint Annual Workshop of the ERCIM Working Group on Constraints and the CoLogNET area on Constraint and Logic Programming », MTA-SZATKI, K. APT, F. FAGES, F. ROSSI, P. SZEREDI, J. VÁNCZA, editors, Budapest, Hungary, July, 2003.

[14] M. DUCASSÉ, L. LANGEVINE, P. DERANSART. *Rigorous Design of Tracers: an Experiment for Constraint Logic Programming.* in « Proceedings of the 5th International Workshop on Automated and Algorithmic Debugging, AADEBUG'03 », Ghent, Belgium, 2003.

[15] F. FAGES. *Symbolic model checking for biochemical systems (invited tutorial).* in « Proceedings of International Conference on Logic Programming ICLP'03 », MIT Press, Mumbai, India, December, 2003.

[16] F. FAGES, A. LAL. *A Global Constraint for Cutset Problems.* in « Proceedings of Fifth International Workshop on Integration of AI and OR techniques in constraint programming for combinatorial optimization problems CPAIOR'03 », M. GENDREAU, G. PESANT, L. ROUSSEAU, editors, Montreal, Canada, April, 2003.

[17] G. FERRAND, W. LESAINT, A. TESSIER. *Towards declarative diagnosis of constraint programs over finite domains.* in « Proceedings of the 5th International Workshop on Automated and Algorithmic Debugging, AADEBUG'03 », M. RONSSE, editor, pages 159–170, Ghent, Belgium, 2003.

[18] L. LANGEVINE, P. DERANSART. *Un nouveau traceur générique pour GNU-Prolog.* in « Programmation en logique avec contraintes, proceedings of JFPLC 2003 », series special issue of RSTI, Hermès, M. DUCASSÉ, editor, pages 97–110, Amiens (France), June, 2003.

[19] L. LANGEVINE, M. DUCASSÉ, P. DERANSART. *A Propagation Tracer for GNU-Prolog: from Formal Definition to Efficient Implementation.* in « Proceedings of ICLP'03, International Conference on Logic Programming », series Lecture Notes in Computer Science, Springer Verlag, C. PALAMIDESSI, editor, Mumbai, India, December, 2003.

Internal Reports

[20] A. BALLESTA. *La détection de symétries dans les problèmes de programmation par contraintes.* Rapport de stage de l'INSA de Rouen, INRIA, September, 2003.

[21] N. CHABRIER, F. FAGES, S. SOLIMAN. *BIOCHAM's user manual.* INRIA, edition 0.2, November, 2003, http://contraintes.inria.fr/BIOCHAM/.

[22] R. COOLEN. *Apport de la réalité virtuelle dans un problème de placement 2D/3D.* Rapport de stage d'option de l'Ecole Polytechnique, INRIA, June, 2003.

[23] E. COQUERY, F. FAGES. *Subtyping constraints in quasi-lattices.* Technical report, number RR-4926, INRIA Rocquencourt, September, 2003, http://www.inria.fr/rrrt/rr-4926.html.

[24] D. GROSU. *Modélisation et interrogation en logique temporelle de processus biochimiques.* Rapport de DEA OMJA Paris, INRIA, September, 2003.

[25] R. HAEMMERLÉ. *Conception et Compilation d'un Langage Noyau Linéaire Concurrent avec Contraintes.* Rapport de DEA SPL Paris, INRIA, September, 2003.

[26] M. PIERRÈS. *Systèmes fortement contraints: émergence évolutionnaire d'un consensus.* Rapport de stage de DEA IARFA, INRIA, September, 2003.

[27] S. SOLIMAN. *Pi-calculus and LCC, a Space Odyssey.* Research Report, number RR-4855, INRIA, June, 2003, http://www.inria.fr/rrrt/rr-4855.html.

[28] N. SZNAJDER. *Introduction de probabilités dans le système BIOCHAM de modélisation de processus biochimiques.* Rapport de MMFAI ENS Paris, INRIA, September, 2003.