INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Contraintes

Constraint Programming

Rocquencourt

THEME SYM

Activity Report

2006

Table of contents

1. Team

Head of project-team
François Fages [Research Director (DR) INRIA]

Vice-head of project-team
Sylvain Soliman [Research Associate (CR) INRIA]

Administrative assistant
Nadia Mesrar [Secretary (SAR) Inria]

Staff member
Pierre Deransart [Research Director (DR) INRIA]

Post-Doctoral fellow
Laurence Calzone [PostDoc APrIL2, up to August 06]
Andras Kovacs [PostDoc ERCIM, from Sep. 06]
Sriram Krishnamachari [PostDoc ARC MOCA, from August 06]

Research scientist (partner)
Nicolas Beldiceanu [Professor, Ecole des Mines de Nantes]
Emmanuel Coquery [Associate Professor, University of Lyon 1]
Daniel Diaz [Associate Professor, University of Paris 1]

Ph.D. students
Nathalie Chabrier-Rivier [INRIA scholarship]
Rémy Haemmerlé [INRIA scholarship]
Julien Martin [INRIA scholarship]
Aurélie Strobbe [CIFRE TOTAL]

Student interns
Adiga Satish [IIT Kanpur]
Jean-Christophe Reussner [ENSTA]

2. Overall Objectives

2.1. Overall Objectives

Constraint Logic Programming supports a great ambition for programming: the one of making of programming essentially a modeling task, with equations, constraints and logical formulas.

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 for solving combinatorial optimization problems in industry or commerce are related to the bringing of new local consistency techniques and of *declarative languages* which allow control on the mixing of heterogeneous resolution techniques: numerical, symbolic, deductive and heuristic.

The "Contraintes" group investigates the logical foundations, design, implementation, programming environments and applications of constraint programming languages. 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 problems and computational system biology. In bioinformatics, our objective is not to work on structural biology problems, which has been the main trend up to now, but to attack the great challenge of systems biology, namely to model the function, activity and interaction of molecular systems in living cells, with logic programming concepts and program verification technologies.

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 = logical 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;
- temporal constraints: CTL and LTL formulae, either propositional or with numerical constraints.

The project works on constraint resolution methods and their cooperation. The main focus is the declarativeness of the constraint solver (e.g. implemented by CHR rules), the efficiency of constraint propagation methods, the design of global constraints and the combination of constraint propagation with local search methods.

3.3. Computational systems biology

Systems biology is a cross-disciplinary domain involving biology, computer science, logics, mathematics, and physics to elucidate the high-level functions of the cell from their biochemical bases at the molecular level.

At the end of the Nineties, research in Bioinformatics evolved, passing from the analysis of the genomic sequence to the analysis of post-genomic interaction networks (expression of RNA and proteins, protein-protein interactions, etc). The complexity of these networks requires a large research effort to develop symbolic notation and analysis tools for biological processes and data. In order to scale-up, and get over the complexity walls to reason about biological systems, there is a general feeling that beyond providing tools to biologists, computer science has much to offer in terms of concepts and methods.

We are interested in the modeling and analysis of complex molecular processes in the cell, at different levels of abstraction, qualitative and quantitative. The most original aspect of our research can be summarized by the following identifications:

biological model = transition system,

biological property = temporal logic formula,

biological validation = model-checking.

Our main research axis is thus the application of logic programming concepts and circuit or program verification techniques to the analysis of complex biochemical processes in the cell.

4. Application Domains

4.1. Combinatorial optimization problems

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

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

The last forty years have brought many improvements in Operations Research resolution techniques. In this context, Constraint Programming can be seen as providing, on the one hand, local consistency 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 the cooperation of global resolution, local consistency techniques and search methods.

The project builds upon its knowledge of CC languages, constraint solvers and their implementation to work in these directions. The LCC paradigm offers at the same time a theoretical framework for analysis, and a valuable guide for practical language design and implementation. The work on programming environments helps to integrate the Constraint Programming tools into this application domain.

4.2. In-silico cell

In 2002 , we started a Collaborative Research Initiative ARC CPBIO on “Process Calculi and Biology of Molecular Networks”. By working on well understood biological models, we sought:

- to identify in the family of competitive models coming from the Theory of Concurrency[1] and from Logic Programming [2], 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.

This work lead us to the design and implementation of the Biochemical Abstract Machine BIOCHAM that has the unique feature of providing formal languages corresponding to different qualitative and quantitative levels of abstraction, for, on the one hand, modeling biomolecular interaction diagrams with reaction rules, and on the other hand, modeling the biological properties of the system in temporal logic. This double formalization of both the model and the biological properties of the system at hand opens several new research avenues on the design and systematic validation of biological models.

We participate in two European Projects of the 6th PCRD. First the STREP APrIL II where the main objective is to apply probabilistic inductive logic programming techniques to bioinformatics applications. Our main focus is on the regulatory and metabolic networks in the cell, and the semi-automatic completion or revision of models from observed temporal properties of the system [3]. Second, the Network of Excellence REWERSE, in which we focus, among other themes, on bioinformatics as a field of application of the new Semantic Web technologies based on rules and constraints. In this context we intend to use the biological knowledge stored in online ontologies like GO in order to improve the re-use and composition of models, as well as the semi-automatic correction/completion of models. In this respect, we developped type inference and abstract interpretation techniques to extract information from reaction models [15].

We coordinate the ARC MOCA (2006-2008) on “MOdularity, Compositionality and Abstraction in gene and protein networks”, and collaborate in this framework with Denis Thieffry, Claudine Chaouyia, Univ. Marseille, Anne Siegel and Ovidiu Radulescu, IRISA SYMBIOSE and IHES, Ralf Blossey IRI and Vincent Danos CNRS Paris, on the exploration of different techniques and methods allowing semi-automatic composition, decomposition, simplification and refinement of biological models. More precisely, we study the formal links between logical and numerical models of some parts of the cell cycle and the decomposition in modules based on control theory intuitions.

This technology developed in the last years is now applied to new biological questions that we investigate in partnerships with biologists in two new projects. First, the EU STREP project TEMPO on “temporal genomics for patient tailored chornotherapeutics”, coordinated by Francis Levi INSERM Villejuif, where, in partnership with Jean Clairambault of the BANG project-team, we develop in BIOCHAM coupled models of the cell cycle, the circadian cycle and the effect of cytotoxic drugs in cancer therapies. Second, the INRA AgroBi project, coordinated by Eric Reiter INRA Tours, where, in partnership with Frédérique Clément of the SOSSO2 project-team, we develop in BIOCHAM models of FSH signaling networks in mammalian cells.

5. Software

5.1. BIOCHAM

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

[1] π-calculus, Join-calculus and their derivatives
[2] Constraint Logic Programming, Concurrent Constraint languages and their extensions to discrete and continuous time, TCC, HCC

The Biochemical Abstract Machine BIOCHAM is a modeling and validation environment for molecular systems biology. BIOCHAM provides precise semantics to biomolecular interaction maps at three abstraction levels:

1. the boolean semantics of molecules presence and absence,
2. the continuous semantics of molecular concentrations,
3. the stochastic semantics of molecule populations.

Based on this formal framework, BIOCHAM offers:

- a compositional rule-based language for modeling biochemical systems, allowing patterns and kinetic expressions when numerical data are available;
- numerical and boolean simulators;
- a temporal logic language (CTL for qualitative models and LTL with numerical constraints for quantitative models) for formalizing biological properties such as reachability, checkpoints, oscillations or stability, and checking them automatically with model-checking techniques;
- automatic search procedures to infer interaction rules and parameter values from known temporal properties of the system.

BIOCHAM is fully implemented in GNU-Prolog and interfaced to the state-of-the-art symbolic model checker NuSMV for evaluating boolean CTL queries in large models over several hundreds of variables. BIOCHAM models can be imported from, and exported to, the standard Systems Biology Markup Language SBML.

5.2. 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. TCLP can also type check constraint solvers written in the *Constraint Handling Rules* (CHR) language.

The flexibility of the TCLP type system is obtained by the combination of 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 a type inference algorithm for predicates and to a default *term* type for function symbols.

5.3. 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, is developed as an extension introducing concurrency, attributed variables, and constraint solving over the 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.

5.4. CLPGUI

Participant: François Fages.

CLPGUI is a generic graphical user interface written in Java for constraint logic programming. It is 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.

5.5. Tra4CP

Participant: Pierre Deransart.

Following the former OADymPPaC project, we have set up with J.-D Fekete a new open source project called Tra4CP (Traces for Constraint Programming). The main objective of this open project is to build a repository of constraint resolution traces and analysis tools.

The purpose is to pursue dissemination the results of the former OADymPPaC project, to facilitate contacts and working exchanges between researchers in the field of complex problem solving using constraint programming, and mainly to promote new analysis methods. Following this idea, the project allowed to integrate recent developments related to the use of the generic trace format Gentra4CP for constraint programming to produce trace analysis based on music (work of Jeremie Vautard Tra4CP). The creation of a working group on constraint problem XML repsesentation and its integration in the Gentra4CP is also considered.

6. New Results

6.1. Implementation of SiLCC

Participants: François Fages, Rémy Haemmerlé, Adiga Satish, Sylvain Soliman.

We are developing SiLCC an imperative and concurrent constraint programming language based on a single paradigm: the one of Vijay Saraswat's concurrent constraint programming extended with constraint systems based on Jean-Yves Girard's Linear Logic. In the late 90's we developed the theory of this extension and we have now begun its implementation.

From a constraint programming point of view, the unique combination of constraint programming with imperative features opens many new possibilities, among which:

- the capability of programming constraint solvers in the language, making them extensible by the user,
- making a fully bootstrapped implementation of a constraint programming language (for the first time since Prolog)
- combining constraint reasoning with state change;
- proving program correctness using Linear Logic.

Our current implementation of SiLCC uses GNU-Prolog-RH as temporary kernel language, on top of which a module system and the first bootstrapping libraries have been developed. The objective is to define a small kernel language as an instance of LCC over a simple constraint domain of labelled graphs, on top of which the complete SiLCC language will be built by bootstrapping. Bootstrapping is a fundamental step for getting over the current limits of today constraint programming tools, concerning their extensibility, robustness, and teaching. The main step realized this year has been the definition of the module system.

6.2. Module System for SiLCC

Participants: François Fages, Rémy Haemmerlé, Sylvain Soliman.

Module systems are an essential feature of programming languages as they facilitate the re-use of existing code and the development of general purpose libraries. However, there has been no consensual module system for Prolog, hence no strong development of libraries, in sharp contrast to what exists in Java for instance. One difficulty comes from the *call* predicate which interferes with the protection of the code, an essential task of a module system. In [16], by distinguishing the called module code protection from the calling module code protection, we have reviewed the existing syntactic module systems for Prolog and shown that no module system ensures both forms of code protection, with the noticeable exceptions of Ciao-Prolog and XSB. We presented a formal module system for logic programs with calls and closures, defined its operational semantics and formally proved the code protection property. Interestingly, we also provided an equivalent logical semantics of modular logic programs without calls nor closures, which shows how they can be translated into constraint logic programs over a simple module constraint system.

The module system of SiLCC goes however beyond the one we propose for Prolog-like languages. There are indeed two somewhat contradictory ways of looking at modules in a given programming language. On the one hand, module systems are largely independent of the particulars of programming languages, and several examples of module systems have indeed been adapted to different programming languages. On the other hand, the module constructs often interfere with the programming constructs, and may be redundant with other scope mechanisms of programming languages, such as closures for instance. There is therefore a need to unify the programming concepts and constructs that are similar, and retain a minimum number of essential constructs to avoid arbitrary programming choices. In [21] we realize this aim in the framework of linear logic concurrent constraint programming (LCC) languages. We first show how declarations and closures can be internalized as agents in LCC. We then present a modular version of LCC (MLCC), where modules are referenced by variables and where implementation hiding is obtained with the usual hiding operator for variables. We develop the logical semantics of MLCC in linear logic, and show the completeness of the operational semantics for the observation of successes and accessible stores.

6.3. Type systems

Participants: Emmanuel Coquery, Pierre Deransart, François Fages, Sylvain Soliman.

The *Constraint Handling Rules* (CHR) are an untyped rewrite rule language, initially designed for implementing constraint solvers. In order to benefit from type programming for the development of larger programs, we have designed a generic type system for CHR [8]. CHR being a high-level extension of a host language, such as Prolog or Java, the type system is parameterized by the type system of the host language. The consistency of the type system has been shown for the operational semantics of CHR, as well as for the extended execution model CLP+CHR, for the case where the host language is a constraint logic programming language. This type system has been implemented as an extension of our type checker TCLP which is itself implemented in CLP+CHR. It has been successfully evaluated on twelve CHR solvers and programs, including TCLP itself.

In [9], an analysis of constraint propagation execution is based on a formal setting of traces and trace drivers that can be tailored to the observation of different types of events.

In the framework of computational systems biology, and in particular for the BIOCHAM modeling environment, we developed type inference and abstract interpretation techniques to either type check reaction models or infer types from reaction models [15]. In this approach the types can represent various information such as the categories of proteins (kinase, phosphtase, etc.) the function of the proteins (activations and inhibitions) or the topology of compartments for instance.

6.4. Constraint programming environments

Participant: Pierre Deransart.

Semantics based debugging, static and dynamic typing, and more generally validation of CC programs are also studied in the project.

We started to study theoretical questions issued from the former OADymPPaC project (RNTL ended in 2004) by trying to clarify the notions of trace exchanged between observed and observing processees and their semantics called "observational semantics". We wanted also to better formalize the interactions between these processes in order to be able to optimize their communications in particular by formalizing the introduction of Tracer Driver and Analyser Manager respectively added to the oberserved and observing processes. Finally the question of extending the domain of application of the approach to other domains than constraints solvers is considered by introducing the concept of "generic obervational semantics" and "generic trace scheme".

This work is the continuation of the work started with Mireille Ducassé and Ludovic langevine on these topics during the OADymPPaC project. First developments have been published in [9].

6.5. Global constraint of scheduling with concentration constraints

Participants: François Fages, Andras Kovacs, Julien Martin, Jean-Christophe Reussner, Aurélie Strobbe.

Global constraints provide a way to introduce redundant constraints from which tighter propagation mechanisms may be infered, for instance by re-using a wide area of algorithms from graph theory. In [7], we describe a global constraint and a filtering algorithm for a pure graph property : computing cutsets in a graph (i.e. sets of vertices to remove to make the graph acyclic). This global constraint is compared to mixed integer programming and applied to a reconciliation problem in nomadic applications, originating from a previous study we did with Microsoft Research Cambridge.

In collaboration with TOTAL, we study an optimization problem of crude oil blending in refineries. The problem is to find a plan of transfer and mixing of crude oil, from the boats to the distillation units through storage and mixing tanks, under various constraints of product concentrations, continuous distillation, schedule, and optimization criteria. This NP-hard problem is the combination of a planning problem (for choosing the transfer tasks and quantities) with a scheduling problem.

Previous approaches based on Mixed-Integer Linear Programming (MILP) rely on a discretization of time that may give infeasible or suboptimal solutions. On the other hand, a constraint programming approach is appealing for its capability to treat efficiently the scheduling part of the problem without time discretization, We have developed such a solution with ILOG tools, by introducing a global constraint of scheduling with concentration constraints, and using heuristics for the planning part of the problem. A filtering algorithm for this global constraint is currently developed as a generalization of the multi-resource cumulative constraint.

Because of the importance of the heuristics part on real-size data, we have also considered a constraint-based local search method for this problem [22]. For this study we used the Comet system of Van Hentenryck developed at Brown University, and a collaboration with Pascal Van Hentenryck is planned on this difficult problem.

Similar methods are also investigated in collaboration with Evelyne Lutton (COMPLEX Team) for multi-user distributed resource allocation problems and for the implementation of the CONSENSUS system.

6.6. Continuous dynamics in BIOCHAM

Participants: Laurence Calzone, François Fages, Sriram Krishnamachari, Sylvain Soliman.

BIOCHAM (the BIOCHemical Abstract Machine) is a software environment for modeling biochemical systems [4], [19]. It is based on two aspects: (1) the analysis and simulation of boolean, differential and stochastic reaction models and (2) the formalization of biological properties in temporal logic. BIOCHAM provides tools and languages for describing protein networks with a straightforward syntax, and for integrating biological properties into the model. It then becomes possible to analyze, query, verify, and maintain the model w.r.t. those properties. For kinetic models, BIOCHAM can search for appropriate parameter values in order to reproduce a specific behavior observed in experiments and formalized in temporal logic. Coupled with other methods such as bifurcation diagrams, this search assists the modeler/biologist in the modeling process.

In the differential semantics, the time series obtained from numerical integration allows us to use model-checking techniques for LTL queries with numerical constraints [3]. A model-checker incorporated in BIOCHAM and implemented in GNU-Prolog, enables the automatic verification of quantifier free first-order LTL formulae (about concentrations and their derivatives) but also of some specific quantified first-order formulae, such as periods of oscillations for instance [18].

Current work involves:

- using this model-checking features for learning parameter values, as explained in section 6.8;
- identifying precisely which fragment of first-order LTL is wanted;
- formalizing, as an abstraction, the relationship between the continuous and boolean semantics;
- defining more precisely the relationship with the stochastic semantics;
- using bifurcation theory to gain information about possible parameter values for certain behaviors of the system, and elucidating how to connect LTL with "types of bifurcations".

6.7. Inferring reaction rules in BIOCHAM

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

The formalization of biomolecular interactions in BIOCHAM syntax, and the specification of the observed behavior of the system in temporal logic (CTL or LTL) make it possible to develop machine learning algorithms to automatically correct or complete existing models. It is worth noting that structural learning of reaction rules from temporal logic properties is quite new, both from the machine learning perspective and from the systems biology perspective.

In the framework of the APrIL II STREP, we first applied state-of-the-art inductive logic programming tools to simple reachability properties. The objective was to complete a boolean BIOCHAM model to satisfy a CTL specification. We developed a more powerful search algorithm for learning new reaction rules from general temporal logic properties. This algorithm evolved into a theory revision algorithm, taking into account that ACTL and ECTL formulae (i.e. those where the only path operator is respectively A and E) can guide the search for the addition or deletion of rules. In the quantitative case, the same strategy can be applied to learn kinetic parameter values. Once the biological properties of a system under various initial conditions, are described both qualitatively and quantitatively with LTL formulae and constraints, an enumerative algorithm can be applied to estimate parameter values by model-checking [3]. This provides an interesting tool to the biologist/modeler who has to proceed in part by trial and error.

These algorithms are currently improved in BIOCHAM by applying various optimizations, such as :

- compressing the simulation trace used by the LTL model checker with constraints for searching parameter values;
- using the type information contained in reaction models [15] to restrict the search for reaction rules to add to a model;
- improving the data structures of BIOCHAM code, in the spirit of its migration from Prolog to SiLCC;
- exploring inductive logic programming techniques, through a translation of Biocham models to Prolog and Stochastic Logic Programs code.

6.8. Coupled model of the cell cycle and the circadian cycle in mammalian cells

Participants: Laurence Calzone, Sriram Krishnamachari, Sylvain Soliman.

Recent advances in cancer chronotherapy techniques support the evidence that there exist some links between the cell and the circadian cycles. Both cycles have been successfully modeled, however, as of today, there are no precise models describing the coupling of the two cycles. One purpose of a coupled model is to better understand how to efficiently target malignant cells depending on the phase of the day. This is at the heart of our participation in the EU STREP project TEMPO.

Our model [18] is built from two models, a model of the circadian cycle from Leloup and Goldbeter and a generic model of the cell cycle model focusing on a crucial event, the entry into mitosis. Currently, the model is focused on the coupling through the WEE1 kinase. This protein plays an important role in the regulation of the activity of MPF, a complex crucial in cell division. The WEE1 kinase seems to be regulated positively by CLOCK/BMAL1 and negatively by CRY, these three genes being well-known circadian core genes.

The BIOCHAM feature for learning parameter values under period constraints was introduced for this study. This allowed us to compute the domain of entrainment of the cell cycle by the circadian cycle. This domain mainly depends on the strength of the WEE1 kinase, regulated by circadian genes, on the activity of MPF. We now intend to incorporate other links, based on evidence from the literature: through the protein c-MYC for instance; and to make the generic cell cycle model more complete in order to see the effect of these other links.

7. Contracts and Grants with Industry

7.1. ILOG

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

7.2. TOTAL

Collaboration with TOTAL on a constraint programming approach to the optimization of crude oil blending and the Thesis of Aurélie Strobbe under a CIFRE contract.

8. Other Grants and Activities

8.1. National contracts

- INRA project AgroBi (2006-2010) on the modeling of FSH signaling, coordinated by Eric Reiter, INRA Tours;
- ARC MOCA (2006-2007) MOdularité, Compositionalité et Abstraction dans les réseaux géniques et protéiques, coordinated by Sylvain Soliman;
- ACI IMPbio VICANNE (2004-2007) Modélisation dynamique et simulation des systèmes biologiques;
- RNTL project MANIFICO (Sep. 2003-2006) on the compilation of rules and constraints. with LORIA PROTHEO and ILOG coord;
- CIFRE thesis and industrial contract with TOTAL (July 2004- July 2007) on the optimization of petroleum processes by constraint programming.

8.2. European contracts

- 6th PCRD STREP Net-WMS (2006-2010) on constraint optimization in Wharehouse Management Systems, ERCIM coord, F. Fages scientific coordinator, Ecole des Mines de Nantes, SICS, KLS optim, CEA, mindbiz, Widescope, PSA, Fiat;
- 6th PCRD STREP TEMPO (2006-2010) on temporal genomics for tailored chronotherapeutics, coordinated by Francis Lévi at INSERM Villejuif;
- 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;
- ERCIM Working Group on Constraints, coord. F. Fages, INRIA Rocquencourt.

8.3. Invitations

Have been invited for short visits :

- Stephanie Spranger, Ludwig Maximilian University, Munich, Germany,
- Bela Novak, Univ. Budapest, Hungary,
- Jacques Robin, Universidade Federal de Pernambuco, Brazil,
- Khalil Djelloul, University of Ulm, Germany.

9. Dissemination

9.1. Teaching

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

Several Ph.D. students and members of Contraintes teach in the first cycles of Universities or Engineering schools. Our involvement in Master courses is the following:

- 24h Course on *Constraint Programming*, Master Parisien de Recherche en Informatique (MPRI) by Sylvain Soliman (12h) and François Fages (6h).
- 48h Course on *Computational Systems Biology*, Master Parisien de Recherche en Informatique (MPRI) by François Fages (15h) and Sriram Krishnamachari (3h).

9.2. Leadership within scientific community

- Pierre Deransart is the General Secretary, past Chairman, of the “Association Française pour la Programmation par Contraintes” AFPC.

  He is "Brazil Correspondent" in the DREI (INRIA international Direction) and organized several INRIA activities related to INRIA-Brazil scientific cooperation.
- François Fages is member of the Editorial Board of RAIRO Operations Research, the Chairman of the ERCIM Working Group on Constraints, vice-Chairman, past Chairman of the “Association Française pour la Programmation par Contraintes” AFPC, member of the Scientific Councils of the Epigenomics project of the Genopole of Evry, and of the French-Russian Liapunov Institute.
- Sylvain Soliman is the Secretary of the ERCIM Working Group on Constraints.

10. Bibliography

Year Publications

Books and Monographs

[1] B. HNICH, M. CARLSSON, F. FAGES, F. ROSSI. *Recent Advances in Constraints, selected papers from the Joint ERCIM/Colognet Workshop CSCLP'05*, Lecture Notes in Artificial Intelligence, vol. 3978, Springer-Verlag, 2006.

Articles in refereed journals and book chapters

[2] L. CALZONE, N. CHABRIER-RIVIER, F. FAGES, S. SOLIMAN. *Langages formels dans la machine abstraite biochimique BIOCHAM*, in "Techniques et Sciences Informatiques", to appear, 2006, http://contraintes.inria.fr/~fages/Papers/CCFS05tsi.pdf.

[3] L. CALZONE, N. CHABRIER-RIVIER, F. FAGES, S. SOLIMAN. *Machine learning biochemical networks from temporal logic properties*, in "Transactions on Computational Systems Biology VI", G. PLOTKIN (editor). , Lecture Notes in BioInformatics, CMSB'05 Special Issue, vol. 4220, Springer-Verlag, November 2006, p. 68–94, http://contraintes.inria.fr/~fages/Papers/CCFS05tcsb.pdf.

[4] L. CALZONE, F. FAGES, S. SOLIMAN. *BIOCHAM: An Environment for Modeling Biological Systems and Formalizing Experimental Knowledge*, in "BioInformatics", vol. 22, n^{o} 14, 2006, p. 1805–1807, http://contraintes.inria.fr/~fages/Papers/bioinformatics.pdf.

[5] F. FAGES. *From Syntax to Semantics in Systems Biology - Towards Automated Reasoning Tools*, in "Transactions on Computational Systems Biology IV", vol. 3939, December 2006, p. 68–70, http://pauillac.inria.fr/~fages/Papers/Fages06tcsb.pdf.

[6] F. FAGES. *Programmation logique et contraintes*, in "Encyclopédie d'informatique, Paris", Vuibert, 2006, p. 151–162.

[7] F. FAGES, A. LAL. *A Constraint Programming Approach to Cutset Problems*, in "Journal Computers and Operations Research", vol. 33:10, October 2006, p. 2852–2865.

Publications in Conferences and Workshops

[8] E. COQUERY, F. FAGES. *A type system for CHR*, in "Recent Advances in Constaints, revised selected papers from CSCLP'05", Lecture Notes in Artificial Intelligence, n^{o} 3978, Springer-Verlag, 2006, p. 100–117, http://pauillac.inria.fr/~fages/Papers/CF05csclp.pdf.

[9] P. DERANSART. *On using Tracer Driver for External Dynamic Process Observation*, in "Workshop on Logic Programming Environments WLPE'06 associated to ICLP'06 and FLOC'06, Seattle, USA", August 2006.

[10] F. FAGES. *Biological validation as model checking*, in "Dagstuhl seminar on Simulation and Verification of Dynamic Systems, Dagstuhl, Germany", In [17], April 2006.

[11] F. FAGES. *Formal Languages in the Biochemical Abstract Machine BIOCHAM*, in "Workshop Logic and Interactions, associated to Geometry of Computation Geocal'06, CIRM, Marseille", February 2006.

[12] F. FAGES. *Machine learning biochemical models from temporal logic properties (tutorial)*, in "April II workshop: Applications of Probabilistic Inductive Logic, associated to ECML'06, Berlin", September 2006.

[13] F. FAGES. *Machine learning biochemical networks from temporal logic properties*, in "Proc. Modélisation de systèmes biologiques complexes dans le contexte de la génomique, Bordeaux", April 2006.

[14] F. FAGES. *On Using Temporal Logic with Constraints to Express Biological Properties of Cell Processes (invited talk)*, in "Proc.Workshop on Constraint Based Methods for Bioinformatics WCB'06, associated to CP'06, Nantes", September 2006, p. 1–5.

[15] F. FAGES, S. SOLIMAN. *Type Inference in Systems Biology*, in "CMSB'06: Proceedings of the fourth international conference on Computational Methods in Systems Biology", C. PRIAMI (editor). , Lecture Notes in Computer Science, vol. 4210, Springer-Verlag, 2006, http://pauillac.inria.fr/~fages/Papers/FS06cmsb.pdf.

[16] R. HAEMMERLÉ, F. FAGES. *Modules for Prolog Revisited*, in "Proceedings of International Conference on Logic Programming ICLP 2006", Lecture Notes in Computer Science, n^{o} 4079, Springer-Verlag, 2006, p. 41–55, http://pauillac.inria.fr/~fages/Papers/HF06iclp.pdf.

[17] D. M. NICOL, C. PRIAMI, H. RIIS-NIELSON, A. M. UHRMACHER. *06161 Abstracts Collection – Simulation and Verification of Dynamic Systems*, in "Simulation and Verification of Dynamic Systems", D. M. NICOL, C. PRIAMI, H. R. NIELSON, A. M. UHRMACHER (editors). , Dagstuhl Seminar Proceedings, n^{o} 06161, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006, http://drops.dagstuhl.de/opus/volltexte/2006/710.

Internal Reports

[18] L. CALZONE, S. SOLIMAN. *Coupling the Cell cycle and the Circadian Cycle*, Research Report, n^{o} 5835, INRIA, February 2006, http://hal.inria.fr/inria-00070191.

[19] N. CHABRIER-RIVIER, F. FAGES, S. SOLIMAN. *BIOCHAM's user manual*, INRIA, 2003–2006, http://contraintes.inria.fr/BIOCHAM/.

[20] R. HAEMMERLÉ, F. FAGES. *Modules for Prolog Revisited*, Technical report, n^{o} RR-5869, INRIA, 2006, http://hal.inria.fr/inria-00070157.

[21] R. HAEMMERLÉ, F. FAGES, S. SOLIMAN. *On Internalizing Modules as Agents in Concurrent Constraint Programming*, Technical report, n^{o} RR-5981, INRIA, 2006, http://hal.inria.fr/inria-00096644/.

[22] J.-C. REUSSNER. *Optimisation de la livraison, du stockage, du mélange et du chargement des pétroles bruts d'une rafinerie*, Rapport de stage de l'ENSTA Paris, INRIA, June 2006.

Miscellaneous

[23] F. FAGES. *Machine learning biochemical networks from temporal logic propertiesJoint April/IQ workshop, Titisee, Germany*, March 2006.