INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team VASY

Validation of Systems

Rhône-Alpes

THEME COM

Contents

VASY *is an* INRIA *project-team gathering scientists from several European countries.* VASY *is based in Grenoble with members also located at Université de Bourgogne, Polytechnic University of Bucharest (Romania), and Saarland University (Germany). Since January 2007, the* VASY *scientists in Grenoble are also members of the* LIG *joint research laboratory of Centre National de Recherche Scientifique, Grenoble* INP*, and Université Joseph Fourier.*

1 Team

Team Leader

Hubert Garavel [DR2 INRIA]

Administrative Assistants

Laetitia Gatier [until June 14, 2008]

Helen Pouchot [since June 16, 2008]

Inria Staff

Frédéric Lang [CR1 INRIA]

Radu Mateescu [CR1 INRIA]

Wendelin Serwe [CR1 INRIA]

Invited Professor

Holger Hermanns [Saarland University and INRIA]

STMicroelectronics Staff

Etienne Lantreibecq

Software Engineers

David Champelovier [until February 18, 2008]

Xavier Clerc [until November 11, 2008]

Yves Guerte

Rémi Hérilier

Romain Lacroix [until December 31, 2008]

Jeanne Merle [since September 1st, 2008]

Louis Paternault [since October 1st, 2008]

Sylvain Robert [until August 8, 2008]

Post-Doctoral Fellows

Claude Helmstetter [since November 17, 2008]

Emilie Oudot [until August 31, 2008]

Olivier Ponsini [until September 30, 2008]

Anton Wijs

PhD Students

Nicolas Coste [STMicroelectronics, Cifre grant]

Jan Stoecker [Cordi grant]

Damien Thivolle [Mesr grant]

Meriem Zidouni [Bull, Cifre grant]

2 Overall Objectives

2.1 Introduction

Created on January 1st, 2000, the Vasy project focuses on formal methods for the design of reliable systems.

We are interested in any system (hardware, software, telecommunication) that comprises *asynchronous concurrency*, i.e., any system whose behavior can be modeled as a set of parallel processes governed by interleaving semantics.

For the design of reliable systems, we advocate the use of formal description techniques together with software tools for simulation, rapid prototyping, verification, and test generation.

Among all existing verification approaches, we focus on *enumerative verification* (also known as *explicit state verification*) techniques. Although less general than theorem proving, these techniques enable an automatic, cost-efficient detection of design errors in complex systems.

Our research combines two main directions in formal methods, the *model-based* and the *language-based* approaches:

- Models provide mathematical representations for parallel programs and related verification problems. Examples of models are automata, networks of communicating automata, Petri nets, binary decision diagrams, boolean equation systems, etc. From a theoretical point of view, research on models seeks for general results, independently of any particular description language.
- In practice, models are often too elementary to describe complex systems directly (this would be tedious and error-prone). Higher level formalisms are needed for this task, as well as compilers that translate high level descriptions into models suitable for verification algorithms.

To verify complex systems, we believe that model issues and language issues should be mastered equally.

2.2 Models and Verification Techniques

By verification, we mean comparison — at some abstraction level — of a complex system against a set of *properties* characterizing the intended functioning of the system (for instance, deadlock freedom, mutual exclusion, fairness, etc.).

Most of the verification algorithms we develop are based on the *labeled transition systems* (or, simply, *automata* or *graphs*) model, which consists of a set of states, an initial state, and a transition relation between states. This model is often generated automatically from high level descriptions of the system under study, then compared against the system properties using various decision procedures. Depending on the formalism used to express the properties, two approaches are possible:

- *Behavioral properties* express the intended functioning of the system in the form of automata (or higher level descriptions, which are then translated into automata). In such a case, the natural approach to verification is *equivalence checking*, which consists in comparing the system model and its properties (both represented as automata) modulo some equivalence or preorder relation. We develop equivalence checking tools that compare and minimize automata modulo various equivalence and preorder relations; some of these tools also apply to stochastic and probabilistic models (such as Markov chains).
- *Logical properties* express the intended functioning of the system in the form of temporal logic formulas. In such a case, the natural approach to verification is *model checking*, which consists in deciding whether the system model satisfies or not the logical properties. We develop model checking tools for a powerful form of temporal logic, the *modal μ-calculus*, which we extend with typed variables and expressions so as to express predicates over the data contained in the model. This extension (the practical usefulness of which was highlighted in many examples) provides for properties that could not be expressed in the standard μ-calculus (for instance, the fact that the value of a given variable is always increasing along any execution path).

Although these techniques are efficient and automated, their main limitation is the *state explosion* problem, which occurs when models are too large to fit in computer memory. We provide software technologies (see § 4.1) for handling models in two complementary ways:

- Small models can be represented *explicitly*, by storing in memory all their states and transitions (*exhaustive* verification);
- Larger models are represented *implicitly*, by exploring only the model states and transitions needed for the verification (*on the fly* verification).

2.3 Languages and Compilation Techniques

Our research focuses on high level languages with an *executable* and *formal* semantics. The former requirement stems from enumerative verification, which relies on the efficient execution of high level descriptions. The latter requirement states that languages lacking a formal semantics are not suitable for safety critical systems (as language ambiguities usually lead

to interpretation divergences between designers and implementors). Moreover, enumerative techniques are not always sufficient to establish the correctness of an infinite system (they only deal with finite abstractions); one might need theorem proving techniques, which only apply to languages with a formal semantics.

We are working on several languages with the above properties:

- LOTOS is an international standard for protocol description (ISO/IEC standard 8807:1989), which combines the concepts of process algebras (in particular CCS and CSP) and algebraic abstract data types. Thus, LOTOS can describe both asynchronous concurrent processes and complex data structures. We use LOTOS for various industrial case studies and we develop LOTOS compilers, which are part of the CADP toolbox (see § 4.1).

- We contributed to the definition of E-LOTOS (*Enhanced*-LOTOS, ISO/IEC standard 15437:2001), a deep revision of LOTOS, which tries to provide a greater expressiveness (for instance, by introducing quantitative time to describe systems with real-time constraints) together with a better user friendliness. Our contributions to E-LOTOS are available on the WEB (see `http://www.inrialpes.fr/vasy/elotos`).

- We are also working on an E-LOTOS variant, named LOTOS NT (LOTOS *New Technology*) [9, 14], in which we can experiment new ideas more freely than in the constrained framework of an international standard. Like E-LOTOS, LOTOS NT consists of three parts: a *data part*, which allows the description of data types and functions, a *process part*, which extends the LOTOS process algebra with new constructs such as exceptions and quantitative time, and *modules*, which provide for structure and genericity. Both languages differ in that LOTOS NT combines imperative and functional features, and is also simpler than E-LOTOS in some respects (static typing, operator overloading, arrays), which should make it easier to implement. We are developing several tools for LOTOS NT: a prototype compiler named TRAIAN (see § 4.2), a translator from (a subset of) LOTOS NT to LOTOS (see § 5.2.2), and an intermediate semantic model named NTIF (*New Technology Intermediate Form*) [5].

2.4 Implementation and Experimentation

As much as possible, we try to validate our results by developing tools that we apply to complex (often industrial) case studies. Such a systematic confrontation to implementation and experimentation issues is central to our research.

3 Application Domains

The theoretical framework we use (automata, process algebras, bisimulations, temporal logics, etc.) and the software tools we develop are general enough to fit the needs of many application domains. They are virtually applicable to any system or protocol made of distributed agents communicating by asynchronous messages. The list of recent case studies performed with the CADP toolbox (see in particular § 5.3) illustrates the diversity of applications:

- *Hardware architectures:* asynchronous circuits, multiprocessor architectures, systems on chip, networks on chip, bus arbitration protocols, cache coherency protocols, hardware/software codesign;
- *Databases:* transaction protocols, distributed knowledge bases, stock management;
- *Consumer electronics:* home networking, video on-demand;
- *Security protocols:* authentication, electronic transactions, cryptographic key distribution;
- *Embedded systems:* smart-card applications, air traffic control, avionic systems;
- *Distributed systems:* virtual shared memory, distributed file systems, election algorithms, dynamic reconfiguration algorithms, fault tolerance algorithms;
- *Telecommunications:* high speed networks, network management, mobile telephony, feature interaction detection;
- *Human-machine interaction:* graphical interfaces, biomedical data visualization;
- *Bioinformatics:* genetic regulatory networks, nutritional stress response, metabolic pathways.

4 Software

4.1 The CADP Toolbox

Participants: David Champelovier, Hubert Garavel [contact person], Rémi Hérilier, Frédéric Lang, Radu Mateescu, Sylvain Robert, Wendelin Serwe, Damien Thivolle.

We maintain and enhance CADP (*Construction and Analysis of Distributed Processes* – formerly known as CÆSAR/ALDÉBARAN *Development Package*) [2], a toolbox for protocols and distributed systems engineering (see `http://www.inrialpes.fr/vasy/cadp`). In this toolbox, we develop the following tools:

- CÆSAR.ADT [12] is a compiler that translates LOTOS abstract data types into C types and C functions. The translation involves pattern-matching compiling techniques and automatic recognition of usual types (integers, enumerations, tuples, etc.), which are implemented optimally.
- CÆSAR [8] is a compiler that translates LOTOS processes into either C code (for rapid prototyping and testing purposes) or finite graphs (for verification purpose). The translation is done using several intermediate steps, among which the construction of a Petri net extended with typed variables, data handling features, and atomic transitions.
- OPEN/CÆSAR [13] is a generic software environment for developing tools that explore graphs on the fly (for instance, simulation, verification, and test generation tools). Such tools can be developed independently of any particular high level language. In this

respect, OPEN/CÆSAR plays a central role in CADP by connecting language-oriented tools with model-oriented tools. OPEN/CÆSAR consists of a set of 16 code libraries with their programming interfaces, such as:

- CAESAR_GRAPH, which provides the programming interface for graph exploration,
- CAESAR_HASH, which contains several hash functions,
- CAESAR_SOLVE, which resolves boolean equation systems on the fly,
- CAESAR_STACK, which implements stacks for depth-first search exploration,
- CAESAR_TABLE, which handles tables of states, transitions, labels, etc.

A number of tools have been developed within the OPEN/CÆSAR environment, among which:

- BISIMULATOR, which checks bisimulation equivalences and preorders,
- DETERMINATOR, which eliminates stochastic nondeterminism in normal, probabilistic, or stochastic systems,
- DISTRIBUTOR, which generates the graph of reachable states using several machines,
- EVALUATOR, which evaluates regular alternation-free μ-calculus formulas,
- EXECUTOR, which performs random execution,
- EXHIBITOR, which searches for execution sequences matching a given regular expression,
- GENERATOR, which constructs the graph of reachable states,
- PROJECTOR, which computes abstractions of communicating systems,
- REDUCTOR, which constructs and minimizes the graph of reachable states modulo various equivalence relations,
- SIMULATOR, XSIMULATOR, and OCIS, which allow interactive simulation, and
- TERMINATOR, which searches for deadlock states.

- BCG (*Binary Coded Graphs*) is both a file format for storing very large graphs on disk (using efficient compression techniques) and a software environment for handling this format. BCG also plays a key role in CADP as many tools rely on this format for their inputs/outputs. The BCG environment consists of various libraries with their programming interfaces, and of several tools, such as:

  - BCG_DRAW, which builds a two-dimensional view of a graph,
  - BCG_EDIT, which allows to modify interactively the graph layout produced by BCG_DRAW,
  - BCG_GRAPH, which generates various forms of practically useful graphs,
  - BCG_INFO, which displays various statistical information about a graph,
  - BCG_IO, which performs conversions between BCG and many other graph formats,
  - BCG_LABELS, which hides and/or renames (using regular expressions) the transition labels of a graph,

  - BCG_MERGE, which gathers graph fragments obtained from distributed graph construction,
  - BCG_MIN, which minimizes a graph modulo strong or branching equivalences (and can also deal with probabilistic and stochastic systems),
  - BCG_STEADY, which performs steady-state numerical analysis of (extended) continuous-time Markov chains,
  - BCG_TRANSIENT, which performs transient numerical analysis of (extended) continuous-time Markov chains, and
  - XTL (*eXecutable Temporal Language*), which is a high level, functional language for programming exploration algorithms on BCG graphs. XTL provides primitives to handle states, transitions, labels, *successor* and *predecessor* functions, etc. For instance, one can define recursive functions on sets of states, which allow to specify in XTL evaluation and diagnostic generation fixed point algorithms for usual temporal logics (such as HML [HM85], CTL [CES86], ACTL [DV90], etc.).

- The connection between explicit models (such as BCG graphs) and implicit models (explored on the fly) is ensured by OPEN/CÆSAR-compliant compilers, e.g.:

  - CÆSAR.OPEN, for models expressed as LOTOS descriptions,
  - BCG_OPEN, for models represented as BCG graphs,
  - EXP.OPEN, for models expressed as communicating automata, and
  - SEQ.OPEN, for models represented as sets of execution traces.

The CADP toolbox also includes additional tools, such as ALDÉBARAN and TGV (*Test Generation based on Verification*) developed by the VERIMAG laboratory (Grenoble) and the VERTECS project-team of INRIA Rennes.

The CADP tools are well-integrated and can be accessed easily using either the EUCALYPTUS graphical interface or the SVL [4] scripting language. Both EUCALYPTUS and SVL provide users with an easy, uniform access to the CADP tools by performing file format conversions automatically whenever needed and by supplying appropriate command-line options as the tools are invoked.

4.2 The TRAIAN Compiler

Participants: David Champelovier, Hubert Garavel [contact person], Yves Guerte, Frédéric Lang.

We develop a compiler named TRAIAN for translating descriptions written in the LOTOS NT

[HM85] M. HENNESSY, R. MILNER, "Algebraic Laws for Nondeterminism and Concurrency", *Journal of the ACM 32*, 1985, p. 137–161.

[CES86] E. M. CLARKE, E. A. EMERSON, A. P. SISTLA, "Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications", *ACM Transactions on Programming Languages and Systems 8*, 2, April 1986, p. 244–263.

[DV90] R. DE NICOLA, F. W. VAANDRAGER, *Action versus State Based Logics for Transition Systems*, *Lecture Notes in Computer Science*, *469*, Springer Verlag, 1990, p. 407–419.

language (see § 2.3) into C programs, which will be used for simulation, rapid prototyping, verification, and testing.

The current version of TRAIAN performs lexical analysis, syntactic analysis, abstract syntax tree construction, static semantics analysis, and C code generation for LOTOS NT types and functions.

Although this version of TRAIAN is still incomplete (it does not handle LOTOS NT processes), it already has useful applications in compiler construction [3]. The recent compilers developed by the VASY project-team — namely AAL, CHP2LOTOS (see § 5.2.3), EVALUATOR 4.0 (see § 5.1.5), EXP.OPEN 2.0 (see § 5.1.7), FSP2LOTOS (see § 5.2.3), LNT2LOTOS (see § 5.2.2), NTIF (see § 2.3), and SVL (see § 5.1.7) — all contain a large amount of LOTOS NT code, which is then translated into C code by TRAIAN.

Our approach consists in using the SYNTAX tool (developed at INRIA Rocquencourt) for lexical and syntactic analysis together with LOTOS NT for semantical aspects, in particular the definition, construction, and traversals of abstract trees. Some involved parts of the compiler can also be written directly in C if necessary. The combined use of SYNTAX, LOTOS NT, and TRAIAN proves to be satisfactory, as regards both the rapidity of development and the quality of resulting compilers.

The TRAIAN compiler can be freely downloaded from the VASY WEB site (see `http://www.inrialpes.fr/vasy/traian`).

5 New Results

5.1 Models and Verification Techniques

5.1.1 The BCG Format and Libraries

Participant: Hubert Garavel, Frédéric Lang, Louis Paternault.

BCG (*Binary-Coded Graphs*) is both a file format for the representation of explicit graphs and a collection of libraries and programs dealing with this format.

In 2008, we continued porting the BCG libraries and programming interfaces to 64-bit architectures. We fixed two bugs in the BCG_INFO tool. We also improved significantly the speed of the BCG iterators, which could be inefficient for large BCG graphs containing many deadlock states. For instance, the execution time for enumerating a graph of 12 million states having 23% deadlock states was reduced from 15 hours to 47 seconds.

We also undertook the development of a new version 2.0 of the BCG format. Indeed, the current version 1.0 of BCG is almost fifteen years old and major changes are needed to fully exploit the capabilities of modern computers. Our work started by experimenting new compression schemes that could be used in BCG 2.0.

In order to assess the effectiveness of these compression schemes, we gathered a collection of 250 very large BCG graphs representing the behaviour of real-life systems, which will eventually enrich the existing VLTS (*Very Large Transition Systems*) benchmark suite[1].

[1] `http://www.inrialpes.fr/vasy/cadp/resources/benchmark\_bcg.html`

5.1.2 The OPEN/CÆSAR Libraries

Participant: Hubert Garavel, Radu Mateescu, Wendelin Serwe, Anton Wijs.

OPEN/CÆSAR is an extensible, modular, language-independent software framework for exploring implicit graphs. This key component of CADP is used to build simulation, execution, verification, and test generation tools.

In 2008, we finished porting the OPEN/CÆSAR libraries and programming interfaces to 64-bit architectures. We revised several hash functions of the CÆSAR_HASH library so as to enhance their speed, dispersion, and portability. We improved in many respects the tables of the CÆSAR_TABLE library so that they can store more elements (up to 2^{34} on 64-bit machines) as well as larger elements; moreover, these tables now have a smaller memory cost and faster access time (up to 2.33 times on certain benchmarks). We modified the sorting algorithms for edge lists so that they yield the same results on architectures having different endianness or using different C compilers.

We also studied *state space caching* techniques for model checking, which help fighting state explosion by keeping, in a memory cache, only a fixed amount of visited states. When the cache is full and a newly encountered state must be stored, some state present in the cache is removed and replaced with the new one. We undertook the development of a generic caching machinery allowing to store arbitrary elements into tree-based hierarchies of caches, each cache being equipped with a particular replacement strategy. We used this caching machinery for breadth-first state space exploration in order to store a fixed amount of the breadth-first levels (a technique we call *sampling*), which reduces memory consumption at the cost of a finite amount of redundant work. Cycle detection is either ensured by constantly increasing the sampling period, i.e., the number of breadth-first levels explored between two subsequent insertions of a level in the cache, or by creating additional caches on the fly when needed.

We proposed several sampling strategies and studied their combinations with hierarchical caches, where each cache may contain a number of levels and where data can be moved from one cache to another. We compared the best strategies found with the plain breadth-first generation performed by the GENERATOR tool of CADP on 35 graphs taken from the VLTS benchmark suite and on the graphs of 5 communication protocols available as CADP demo examples. These experiments showed significant reductions of the number of states stored in memory (by a factor ranging from 1.4 up to 36) and of the execution time (up to a factor 4).

Finally, we experimented the use of hierarchical caches also in conjunction with depth-first state space exploration, showing reductions by 70% of the number of states stored in memory and a negligible increase of the execution time compared with standard depth-first exploration. This work was accepted for publication in an international conference [33].

5.1.3 The CÆSAR_SOLVE Library

Participant: Hubert Garavel, Yves Guerte, Rémi Hérilier, Radu Mateescu, Emilie Oudot, Sylvain Robert.

CÆSAR_SOLVE is a generic software library for solving boolean equation systems of alternation depth 1 (i.e., without mutual recursion between minimal and maximal fixed point equations)

on the fly. This library is at the core of several CADP verification tools, namely the equivalence checker BISIMULATOR (see § 5.1.4), the model checkers EVALUATOR 3.5 and 4.0 (see § 5.1.5), and the minimization tool REDUCTOR. The resolution method is based on boolean graphs, which provide an intuitive representation of dependencies between boolean variables, and which are handled implicitly, in a way similar to the OPEN/CÆSAR interface [13].

The CÆSAR_SOLVE library provides seven different resolution algorithms named A1 to A7. A1 and A2 are general algorithms based upon depth-first, respectively breadth-first, traversals of boolean graphs. A3 and A4, based upon memory-efficient depth-first traversals of boolean graphs, are optimized for the case of acyclic, respectively disjunctive/conjunctive, boolean graphs. A5 is a general algorithm based upon a depth-first traversal of boolean graphs; it generalizes Tarjan's algorithm for computing strongly connected components and is much faster than A1 and A2 when it is invoked many times on the same equation block. A6 and A7, based upon memory-efficient breadth-first traversals of boolean graphs, are optimized for the case of disjunctive minimal fixed point (respectively, conjunctive maximal fixed point) equation blocks on which a single resolution is requested. All these algorithms can generate diagnostics (examples and counterexamples) explaining why a result is true or false.

In 2008, we enhanced the CÆSAR_SOLVE library (16,500 lines of C code) by adding a new resolution algorithm, named A8, based upon a depth-first search of the dependency graph between boolean variables. This algorithm is different from the other depth-first search based resolution algorithms of CÆSAR_SOLVE (A1, A3, A4, A5) in the sense that upon traversing a dependency between two boolean variables x and y, the exploration of the remaining "neighbour" dependencies (i.e., transitions going out from x in the boolean graph) is suspended until the information provided by the traversed dependency is fully exploited. If this information allows to establish the truth value of x, then the remaining dependencies from x are discarded; otherwise, the exploration of these dependencies is resumed when the value of x is again needed during resolution. This suspend-resume depth-first search makes A8 more memory-efficient than the other depth-first search based algorithms of CÆSAR_SOLVE when a single outgoing dependency is sufficient for determining the truth value of boolean variables. In practice, algorithm A8 is particularly appropriate for solving the boolean equation systems encoding the comparison between graphs that contain a high degree of nondeterminism and are equivalent modulo the relation considered. This work led to a short paper [28] and an article [29] published in international conferences.

In addition to correcting three bugs in CÆSAR_SOLVE, we ported this library, together with the prototype parallel resolution algorithm proposed in [JM06], to 64-bit architectures. The C code of both CÆSAR_SOLVE and the prototype parallel algorithm was modified to remove all warnings generated by recent compilers and code checkers (namely, GCC, INTEL's ICC, and SUN's CC and LINT) running in 64-bit mode.

We also continued the testing and validation of CÆSAR_SOLVE by adding new examples of boolean equation systems (either represented explicitly as compressed text files, or implicitly as random configuration files) to our non-regression database, which contains now 19,400 examples. The testing of CÆSAR_SOLVE is carried out by shell scripts, which invoke the BES_SOLVE

[JM06] C. JOUBERT, R. MATEESCU, "Distributed On-the-Fly Model Checking and Test Case Generation", *in : Proceedings of the 13th International SPIN Workshop on Model Checking of Software SPIN'2006 (Vienna, Austria)*, A. Valmari (editor), *Lecture Notes in Computer Science*, *3925*, Springer Verlag, p. 126–145, March–April 2006.

tool on each example of boolean equation system, check the compatibility of the results across various machine architectures (32-bit and 64-bit) and various resolution algorithms (from A0 to A8), and also compare the sequential resolution algorithms of CÆSAR_SOLVE with the prototype parallel resolution algorithm.

5.1.4 The BISIMULATOR Tool

Participants: Hubert Garavel, Radu Mateescu, Emilie Oudot, Sylvain Robert.

BISIMULATOR is an equivalence checker that takes as input two graphs to be compared (one represented implicitly using the OPEN/CÆSAR environment, the other represented explicitly as a BCG file) and determines whether they are equivalent (modulo a given equivalence relation) or whether one of them is included in the other (modulo a given preorder relation). BISIMULATOR works on the fly, meaning that only those parts of the implicit graph pertinent to verification are explored. Due to the use of OPEN/CÆSAR, BISIMULATOR can be applied directly to descriptions written in high level languages (for instance, LOTOS). This is a significant improvement compared to older tools (such as ALDÉBARAN and FC2IMPLICIT) which only accepted lower level models (networks of communicating automata).

BISIMULATOR works by reformulating the graph comparison problem in terms of a boolean equation system, which is solved on the fly using the CÆSAR_SOLVE library (see § 5.1.3). A useful functionality of BISIMULATOR is the generation of a "negative" diagnostic (i.e., a counterexample), which explains why two graphs are not equivalent (or not included one in the other). The diagnostics generated by BISIMULATOR are directed acyclic graphs and are usually much smaller than those generated by other tools (such as ALDÉBARAN) that can only generate counterexamples restricted to sets of traces.

In 2008, we continued the development of BISIMULATOR (16,300 lines of C code) by devising new encodings of observational equivalence and branching equivalence in terms of boolean equation systems. These encodings stem from the observation that, on graphs without τ-cycles, transitive reflexive closures over τ-transitions can be computed by using maximal fixed point boolean equations (instead of minimal fixed point ones as in the general case), which can be incorporated into the maximal fixed point equation block encoding the equivalence relation. The elimination of τ-cycles preserves both observational and branching equivalence and can be performed on the fly by applying the τ-compression algorithms available in CADP [16].

We enhanced the BISIMULATOR tool with a prototype implementation of this new encoding of branching equivalence, coupled with the new boolean resolution algorithm A8 (see § 5.1.3). The experiments we conducted indicate that the new encoding brings significant performance improvements with respect to the existing one: for example, when comparing the graph of Philips' Bounded Retransmission Protocol (for 12 retransmissions and messages of length 20) against the graph of its service modulo branching equivalence, the execution time dropped from 441 to 5 seconds and the memory consumption decreased from 174 to 31 Megabytes. This work led to a short paper [28] and an article [29] published in international conferences. Additionally, we continued the systematic non-regression testing and semantical checking of BISIMULATOR.

5.1.5 The EVALUATOR 3.5 and 4.0 Tools

Participants: David Champelovier, Hubert Garavel, Radu Mateescu, Sylvain Robert, Damien Thivolle.

EVALUATOR is a model checker that evaluates a temporal logic property on a graph represented implicitly using the OPEN/CÆSAR environment. Properties are described in regular alternation-free μ-calculus, a logic built from boolean operators, possibility and necessity modalities containing regular expressions denoting transition sequences, and fixed point operators without mutual recursion between least and greatest fixed points. The input language of the tool also allows to define parameterized temporal operators and to group them into separate libraries. EVALUATOR works on the fly, meaning that only those parts of the implicit graph pertinent to verification are explored. The model checking problem is reformulated in terms of solving a boolean equation system. A useful feature of EVALUATOR is the generation of diagnostics (examples and counterexamples) explaining why a formula is true or false.

In 2008, we enhanced the modal equation systems serving as intermediate language for the EVALUATOR 3.5 tool with a special form of equation block specifying fairness properties. This new type of equation block represents the infinite looping operator of PDL-Δ [Str82], which states the existence of an infinite transition sequence made by concatenation of regular subsequences. This extension allows to translate temporal logics involving fairness operators, such as the CTRL logic (see below) directly into the intermediate language of EVALUATOR 3.5, which is closer to boolean equation systems and therefore provides a tighter connection to the verification engine CÆSAR_SOLVE (see § 5.1.3). Additionally, we ported the EVALUATOR 3.5 tool to 64-bit architectures.

We also continued the development of the EVALUATOR 4.0 prototype tool (4,800 lines of SYNTAX code, 37,000 lines of LOTOS NT code, and 8,000 lines of C code), which accepts as input specifications written in MCL (*Model Checking Language*), an extension of the regular alternation-free μ-calculus of EVALUATOR 3.5 with data-handling and fairness operators. In addition to fixing several bugs and porting EVALUATOR 4.0 to 64-bit architectures, we enhanced the translation of MCL formulas into parameterized modal equation systems in order to reduce the size of equation blocks, which directly influences the verification time. This enhancement increased the execution speed of EVALUATOR 4.0 by a factor of two with respect to EVALUATOR 3.5. We also performed non-regression testing between the two versions of the tool: on 30,630 verification runs involving formulas of regular alternation-free μ-calculus, we observed that both versions yield the same verification results and all diagnostics generated are included in the graphs under verification modulo the preorder of strong bisimulation. A paper on EVALUATOR 4.0 was published in an international conference [32].

Also, in the framework of the EC-MOAN project (see § 6.1), we defined the syntax and semantics of CTRL (*Computation Tree Regular Logic*), an extension of CTL [CES86] with regular expressions and fairness operators facilitating the description of biologically-relevant properties. We also proposed a translation from CTRL to HMLR (*Hennessy-Milner Logic with Recur-*

[Str82] R. STREETT, "Propositional Dynamic Logic of Looping and Converse", *Information and Control*, 54, 1982, p. 121–141.

[CES86] E. M. CLARKE, E. A. EMERSON, A. P. SISTLA, "Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications", *ACM Transactions on Programming Languages and Systems 8*, 2, April 1986, p. 244–263.

sion) [Lar88], which is accepted as input by EVALUATOR 3.5. This translation was implemented by the tool CTRL2BLK, which was developed in collaboration with Estelle Dumas, Hidde de Jong, Pedro Monteiro, and Michel Page (IBIS project-team) using the SYNTAX/TRAIAN compiler construction technology advocated by VASY. Used in conjunction with EVALUATOR 3.5, the CTRL2BLK translator provides an on the fly model checker for CTRL formulas on the state/transition graphs generated by the GNA (*Genetic Network Analyzer*) tool developed by IBIS. CTRL and its model checker were employed for analyzing the dynamic behaviour of a genetic regulatory network controlling the carbon starvation response of *Escherichia Coli*. This work led to a publication in an international conference [27].

5.1.6 The XTL Tool

Participants: Hubert Garavel, Radu Mateescu.

XTL (*eXecutable Temporal Language*, see § 4.1) is both a meta-language and tool allowing to specify temporal logic properties involving data values and to verify them on graphs represented in the BCG format.

In 2008, in addition to fixing a bug and porting XTL to 64-bit architectures, we enhanced the language and the tool as follows:

- We introduced a new clause "**any** *T*", which denotes a placeholder for a variable of type *T* in the declarations of anonymous tuples occurring in the "**let**" expressions. This feature allows to simplify XTL programs by avoiding spurious variable declarations.

- We implemented a new overloaded function "**replace**", which takes two arguments having the same predefined XTL type and returns the value of the second argument without evaluating the first one. Used in conjunction with XTL iteration expressions, this function allows to specify arbitrary assignments of accumulator variables during iterations. We updated accordingly the two XTL libraries implementing an ACTL model checker [Pec98].

- We added a new XTL expression "**use**" specifying that a set of variables previously defined in the XTL program are used. We updated several XTL libraries and programs of the CADP demonstration examples with the "**use**" expression wherever appropriate, which improved the quality of the C code generated by XTL and also allowed to detect a few occurrences of variables defined but never used.

- At BULL's request, we developed a new XTL library for computing the radius of a graph (i.e., the maximal distance from the initial state to any other state of the graph) encoded in the BCG format. This information is useful for limiting the depth of graph explorations performed by various tools of CADP, such as EXHIBITOR or TERMINATOR.

[Lar88] K. G. LARSEN, "Proof Systems for Hennessy-Milner Logic with Recursion", *in: Proceedings of the 13th Colloquium on Trees in Algebra and Programming CAAP'88 (Nancy, France)*, M. Dauchet, M. Nivat (editors), *Lecture Notes in Computer Science, 299*, Springer Verlag, p. 215–230, Berlin, March 1988.

[Pec98] C. PECHEUR, "Advanced Modelling and Verification Techniques Applied to a Cluster File System", *Research Report number RR-3416*, INRIA, Grenoble, May 1998.

5.1.7 Compositional Verification Tools

Participants: Rémi Hérilier, Frédéric Lang, Radu Mateescu.

The CADP toolbox contains various tools dedicated to compositional verification, among which EXP.OPEN 2.0, PROJECTOR 3.0, and SVL play a central role. EXP.OPEN 2.0 explores on the fly the graph corresponding to a network of communicating automata (represented as a set of BCG files). PROJECTOR 3.0 implements behavior abstraction [GSL96,KM97] by taking into account interface constraints. SVL (*Script Verification Language*) is both a high level language for expressing complex verification scenarios and a compiler dedicated to this language.

In 2008, we corrected a few minor bugs in these tools, we ported them to 64-bit architectures, and we enhanced them along the following lines:

- At BULL's request, we made two improvements in EXP.OPEN 2.0. First, the partial order reduction preserving stochastic branching equivalence now includes the effect of the partial order reduction preserving branching equivalence, which also preserves stochastic branching equivalence. Second, we extended the input language of EXP.OPEN 2.0 so that a network of communicating automata described in an EXP.OPEN file may now include other EXP.OPEN files recursively, thus allowing the description of large networks of automata in separate files.
- We compared PROJECTOR 2.0 and PROJECTOR 3.0 on about 92,000 automatic tests and found a few tests in which PROJECTOR 3.0 ran slower (up to 6 times) than PROJECTOR 2.0. This performance problem was due to collisions caused by a hash function, which we thus replaced. PROJECTOR 3.0 now runs always faster (3 times on average) than PROJECTOR 2.0 on all tests. More precisely, on 91,840 tests generated randomly, PROJECTOR 3.0 runs 3.3 times faster and uses 1.5 times less memory than PROJECTOR 2.0, whereas on 28 realistic examples, PROJECTOR 3.0 runs 4.3 times faster and uses 3.2 times less memory than PROJECTOR 2.0.
- We added to SVL a new operator "**cut**", which eliminates from a graph all transitions whose label matches a user-given regular expression. SVL implements the "**cut**" operator by calling the EXP.OPEN 2.0 tool.

For the purpose of automated cross-testing among different architectures, we also brought changes to EXP.OPEN 2.0, PROJECTOR 3.0, and SVL so that each of these tools produces exactly the same output on all architectures supported by CADP.

We also designed a technique that consists in finding confluent transitions (not necessarily τ-transitions) in the BCG graphs of a network of communicating automata to identify transitions that are τ-confluent in the product graph. Following known results on τ-confluence, these

[GSL96] S. GRAF, B. STEFFEN, G. LÜTTGEN, "Compositional Minimization of Finite State Systems using Interface Specifications", *Formal Aspects of Computation 8*, 5, September 1996, p. 607–616.

[KM97] J.-P. KRIMM, L. MOUNIER, "Compositional State Space Generation from LOTOS Programs", *in: Proceedings of TACAS'97 Tools and Algorithms for the Construction and Analysis of Systems (University of Twente, Enschede, The Netherlands)*, E. Brinksma (editor), *Lecture Notes in Computer Science*, *1217*, Springer Verlag, Berlin, April 1997. Extended version with proofs available as Research Report VERIMAG RR97-01.

transitions can be given priority in the product graph, thus allowing to generate a smaller graph while preserving branching equivalence. We started a series of experiments to study the potential benefits of this technique.

A paper on the automatic generation of refined interfaces for compositional verification was published in an international conference [37].

5.1.8 Other Tool Developments

Participants: David Champelovier, Hubert Garavel, Yves Guerte, Rémi Hérilier, Romain Lacroix, Frédéric Lang, Radu Mateescu, Jeanne Merle, Louis Paternault, Sylvain Robert, Wendelin Serwe, Damien Thivolle.

A key objective for the future of CADP is the ability to support recent computing platforms. This is a heavy task because of the number of tools in CADP, their intrinsic complexity, and their reliance upon third-party software. In 2008, we made significant steps in this direction:

- As regards 32-bit machines, we brought support for WINDOWS VISTA, MACOS 10.5, and the latest LINUX distributions (DEBIAN, FEDORA CORE, RED HAT, and UBUNTU).
- As regards 64-bit machines, we completed porting CADP to SPARC V9 stations running SOLARIS 10, to INTEL IA64 machines running LINUX, and to INTEL EMT64/AMD64 machines running LINUX.
- We enhanced the C code generated by CADP tools so that it compiles without any warning message using various compilers (namely GCC 3 and GCC 4, SUN's CC, and INTEL's ICC).
- We pursued our collaboration with Pierre Boullier, Philippe Deschamp, and Benoît Sagot (ALPAGE project-team of INRIA Rocquencourt) to port the SYNTAX compiler generation software (more than 300,000 lines of C code) to 12 different platforms (processor, operating system, and C compiler), and especially to 64-bit architectures.

  We introduced prototype declarations for all C functions of SYNTAX, which enabled stricter type checking for function calls. We discovered 18 bugs, 15 of which have been repaired in collaboration with the ALPAGE project-team. We checked (using debugging tools such as VALGRIND and MALLOCSCRIBBLE) that the SYNTAX code is free from faulty memory accesses. We wrote an installation guide for SYNTAX and migrated all CADP tools to the latest version of SYNTAX.

Because of the growing usage of CADP in industry and academy, we pursued our efforts to master the software quality of CADP:

- We continued improving the source code of CADP tools, as well as the C code generated dynamically by CADP tools, so as to suppress warning messages issued by recent C compilers and code-checkers (such as LINT), even with the strictest code-checking options and on all supported computing platforms. We also checked (using MALLOCSCRIBBLE) that this generated C code is free from faulty memory accesses.

- We continued building a comprehensive validation framework, based on non-regression testing and semantical checking for the CADP tools. This framework allows functional testing of individual tools as well as integration testing for several CADP tools used altogether to perform complex verification scenarios on various computing platforms and using various compilers.
- We enriched this framework to obtain statistical information about computing resources (memory and CPU time) consumed during testing, which provides valuable insights about performance evolutions of the CADP tools.
- We continued gathering large collections of benchmarks (BCG graphs, boolean equation systems, μ-calculus formulas, etc.) for testing the CADP tools extensively.
- For the development of CADP, we started experimenting the prototype PIPOL platform of INRIA, which provides reservation, configuration, and deployment facilities for porting and testing applications across various hardware and software environments.

We also rewrote the CADP_MEMORY and PREDICTOR tools in order to obtain better estimations of memory usage across 32-bit and 64-bit platforms.

We experimented the use of various ECLIPSE facilities to develop graphical editors for textual languages. In particular, we developed two syntactic editors for the LUSTRE synchronous language and the μ-calculus language used as input by EVALUATOR 3.5. These prototype editors provide desirable features, such as coloring, error highlighting, undo/redo mechanisms, etc.

Other research teams took advantage of the software components provided by CADP (e.g., the BCG and OPEN/CÆSAR environments) to build their own research software. We can mention the following developments:

- the CTTOOL for the verification of JDC (*Java Distributed Component*) specifications [AAB+07,CCH+08], developed at the Universities of Valparaiso and Santiago (Chile) and at INRIA Sophia-Antipolis;
- the ADAPTOR tool [CPS08] for model-based adaptation of behavioral mismatching components, developed at the University of Málaga (Spain) and at Université d'Evry (France);
- a static analysis tool [AFJV08] developed at the University of Málaga (Spain);

[AAB+07] S. AHUMADA, L. APVRILLE, T. BARROS, A. CANSADO, E. MADELAINE, E. SALAGEANU, "Specifying Fractal and GCM Components with UML", *in: Proceedings of the XXVI International Conference of the Chilean Computer Science Society SCCC'2007 (Iquique, Chile)*, H. Astudillo, E. Tanter (editors), IEEE Computer Society Press, p. 53–62, November 2007.

[CCH+08] A. CANSADO, D. CAROMEL, L. HENRIO, E. MADELAINE, M. RIVERA, E. SALAGEANU, "A Specification Language for Distributed Components Implemented in GCM/ProActive", *in: The Common Component Modeling Example: Comparing Software Component Models CoCoME'2007 (Dagstuhl, Germany)*, A. Rausch, R. Reussner, R. Mirandola, F. Plasil (editors), *Lecture Notes in Computer Science*, *5153*, Springer Verlag, p. 418–448, August 2008.

[CPS08] C. CANAL, P. POIZAT, G. SALAÜN, "Model-Based Adaptation of Behavioral Mismatching Components", *IEEE Transactions on Software Engineering 34*, 4, August 2008, p. 546–563.

[AFJV08] M. ALPUENTE, M. FELIÚ, C. JOUBERT, A. VILLANUEVA, "Using Datalog and Boolean Equation

- the COSTO tool [AAA08] for analyzing KMELIA components and services, developed at Université de Nantes (France);
- the MOTOR tool environment [BHK07] underlying the MODEST language, developed at the University of Saarbrücken (Germany);
- the ARCADE environment [BCH+08a,BCH+08b] for analyzing input/output Interactive Markov Chains, developed at the University of Saarbrücken (Germany) and the University of Twente (The Netherlands);
- the CORAL tool [BCS08] for analyzing availability of systems, developed at the University of Saarbrücken (Germany) and the University of Twente (The Netherlands);
- tools for random exploration and testing [Oud07,DGG+08] developed at Université Paris Sud (France);
- tools for interoperability test case generation [DV07] developed at Université de Rennes (France);

Systems for Program Analysis", *in : Proceedings of the 13th International Workshop on Formal Methods for Industrial Critical Systems FMICS'2008 (L'Aquila, Italy)*, D. Cofer, A. Fantechi (editors), *Lecture Notes in Computer Science*, Springer Verlag, September 2008.

[AAA08] P. ANDRÉ, G. ARDOUREL, C. ATTIOGBÉ, "Composing Components with Shared Services in the Kmelia Model", *in : Proceedings of the 7th International Symposium on Software Composition SC'2008 (Budapest, Hungary)*, C. Pautasso, E. Tanter (editors), *Lecture Notes in Computer Science*, *4954*, Springer Verlag, p. 125–140, March 2008.

[BHK07] H. BOHNENKAMP, H. HERMANNS, J.-P. KATOEN, "Motor: The Modest Tool Environment", *in : Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2007 (Braga, Portugal)*, O. Grumberg, M. Huth (editors), *Lecture Notes in Computer Science*, *4424*, Springer Verlag, p. 500–504, March 2007.

[BCH+08a] H. BOUDALI, P. CROUZEN, B. R. HAVERKORT, M. KUNTZ, M. STOELINGA, "Arcade – A Formal, Extensible, Model-Based Dependability Evaluation Framework", *in : Proceedings of the 13th IEEE International Conference on Engineering of Complex Computer Systems ICECCS'2008 (Belfast, Northern Ireland)*, K. Breitman, J. Woodcock, R. Sterritt, M. Hinchey (editors), IEEE Computer Society Press, p. 243–248, March 2008.

[BCH+08b] H. BOUDALI, P. CROUZEN, B. R. HAVERKORT, M. KUNTZ, M. STOELINGA, "Rich Interfaces for Dependability: Compositional Methods for Dynamic Fault Trees and Arcade models", *in : Proceedings of the 2nd International Workshop on Foundations of Interface Technologies FIT'2008 (Budapest, Hungary)*, K. G. Larsen, A. Wasowski, U. Nyman (editors), April 2008.

[BCS08] H. BOUDALI, P. CROUZEN, M. STOELINGA, "CORAL – A Tool for Compositional Reliability and Availability Analysis", *in : Proceedings of the ARTIST workshop: Tool Platforms for Embedded Systems Modeling, Analysis and Validation. Satellite event of the 19th International Conference on Computer Aided Verification CAV'2007 (Berlin, Germany)*, July 2008.

[Oud07] J. OUDINET, "Uniform Random Walks in Very Large Models", *in : Proceedings of the 2nd International Workshop on Random Testing RT'2007 (Atlanta, Georgia, USA)*, M.-C. Gaudel, J. Mayer, R. Merkel (editors), ACM Computer Society Press, p. 26–29, November 2007.

[DGG+08] A. DENISE, M.-C. GAUDEL, S.-D. GOURAUD, R. LASSAIGNE, J. OUDINET, S. PEYRONNET, "Coverage-Biased Random Exploration of Large Models and Application to Testing", *Rapport de Recherche number 1494*, CNRS-Université Paris Sud / LRI, June 2008.

[DV07] A. DESMOULIN, C. VIHO, "Automatic Interoperability Test Case Generation based on Formal Definitions", *in : 12th International Workshop on Formal Methods for Industrial Critical Systems FMICS'2007 (Berlin, Germany)*, S. Leue, P. Merino (editors), *Lecture Notes in Computer Science*, *4916*, Springer Verlag, p. 234–250, July 2007.

- the TURTLE tool [FMdSSV07] for analyzing real-time specifications, developed at Université de Toulouse (France);
- the C.OPEN tool for analyzing C code [dMGMS07] developed at the University of Málaga (Spain);
- a framework for performance evaluation and functional verification in stochastic process algebras [HMS08] developed at the University of Tehran (Iran) and the University of Eindhoven (The Netherlands);
- an enhanced version of the ETI electronic tool integration platform [MS07] developed at the Universities of Potsdam and Dortmund (Germany);
- the tools CHARMY and SYNTHESIS for analyzing software architectures [Muc07,PTBP08] developed at the Universities of l'Aquila and Camerino (Italy) and at the IMT Lucca Institute for Advanced Studies (Italy);
- the DCOMPOSITOR tool for generating service wrapper protocols [Sal08] developed at the University of Málaga (Spain);
- tools for the automatic generation of interfaces for model checking [SP06] developed at the University of Malta;

[FMdSSV07] B. FONTAN, S. MOTA, P. DE SAQUI-SANNES, T. VILLEMUR, "Temporal Verification in Secure Group Communication System Design", *in : Proceedings of the International Conference on Emerging Security Information, Systems and Technologies SECURWARE'2007 (Valencia, Spain)*, J. Mulholland, O. Nieto-Taladriz (editors), IEEE Computer Society Press, p. 175–180, October 2007.

[dMGMS07] M. DEL MAR GALLARDO, P. MERINO, D. SANÁN, "Extending CADP for Analyzing C Code", *in : Proceedings of the 5th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems MSVVEIS'2007 (Funchal, Madeira, Portugal)*, J. C. Augusto, J. Barjis, U. Ultes-Nitsche (editors), INSTICC Press, p. 104–113, June 2007.

[HMS08] H. HOJJAT, M. MOUSAVI, M. SIRJANI, "A Framework for Performance Evaluation and Functional Verification in Stochastic Process Algebras", *in : Proceedings of the 23rd Annual ACM Symposium on Applied Computing SAC'08 (Fortaleza, Ceará, Brazil)*, R. L. Wainwright, H. Haddad (editors), ACM Computer Society Press, p. 339–346, March 2008.

[MS07] T. MARGARIA, B. STEFFEN, "LTL Guided Planning: Revisiting Automatic Tool Composition in ETI", *in : Proceedings of the 31st IEEE/NASA Software Engineering Workshop SEW'2007 (Columbia, USA)*, IEEE Computer Society Press, p. 214–226, March 2007.

[Muc07] H. MUCCINI, "Using Model Differencing for Architecture-Level Regression Testing", *Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications SEAA'2007 (Lübeck, Germany)*, August 2007, p. 59–66.

[PTBP08] P. PELLICCIONE, M. TIVOLI, A. BUCCHIARONE, A. POLINI, "An Architectural Approach to the Correct and Automatic Assembly of Evolving Component-Based Systems", *Journal of Systems and Software 81*, 12, December 2008, p. 2237–2251.

[Sal08] G. SALAÜN, "Generation of Service Wrapper Protocols from Choreography Specifications", *in : Proceedings of the 6th IEEE International Conference on Software Engineering and Formal Methods SEFM'2008 (Cape Town, South Africa)*, A. Cerone, S. Gruner (editors), IEEE Computer Society Press, p. 313–322, November 2008.

[SP06] S. SPINA, G. PACE, "Automatic Interface Generation for Enumerative Model Checking", *in : Proceedings of the 4th Computer Science Annual Workshop CSAW'2006 (Msida, Malta)*, J. Abela, A. Dalli, K. Guillaumier (editors), University of Malta, p. 116–122, December 2006.

- the SynthesisRt tool for generating adaptors for real-time components [TFGG07], developed at the University of l'Aquila (Italy) together with the PopArt project-team of Inria;

- an enhanced version of the Cress tool [TT07] for developing composed grid services, developed at the University of Stirling (United Kingdom);

- tools for analyzing Web service descriptions [MTO+05,TC05] developed at the University of Montreal (Canada).

5.2 Languages and Compilation Techniques

5.2.1 Compilation of LOTOS

Participants: David Champelovier, Hubert Garavel, Wendelin Serwe.

The Cadp toolbox contains several tools dedicated to the Lotos language, namely the Cæsar.adt compiler [12] for the data type part of Lotos, the Cæsar compiler [8] for the process part of Lotos, and the Cæsar.indent pretty-printer.

In 2008, we migrated those compilers to the latest version of Syntax (see § 5.1.8). We also brought support for long Lotos or C identifiers, as well as for large Lotos programs having more than 2^{16} lines. We ported to 64-bit platforms the C code generated by Cæsar and Cæsar.adt, and we modified this code so as to remove all warnings generated by recent compilers and code checkers (namely, Gcc, Intel's Icc, and Sun's Cc and Lint) running in 32-bit and 64-bit mode.

By applying intensive testing, we discovered and fixed several difficult semantic issues in Cæsar's optimizations that would, in some rare cases, generate incorrect or suboptimal code.

We implemented in Cæsar.adt a feature (already present in Traian since 2003) that allows values of dynamic data types (such as lists, trees, etc.) to be represented "canonically", meaning that they are stored in tables and, thus, represented only once in memory. A technical challenge was to make this feature optional, in the sense that Cæsar.adt users can do this selectively, for a set of types of their choice, while other types remain implemented as before.

[TFGG07] M. Tivoli, P. Fradet, A. Girault, G. Gössler, "Adaptor Synthesis for Real-Time Components", *in: Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2007 (Braga, Portugal)*, O. Grumberg, M. Huth (editors), *Lecture Notes in Computer Science*, *4424*, Springer Verlag, p. 185–200, April 2007.

[TT07] K. L. L. Tan, K. J. Turner, "Automated Analysis and Implementation of Composed Grid Services", *in: Proceedings of the 3rd South-East European Workshop on Formal Methods (Thessaloniki, Greece)*, D. Dranidis, I. Sakellariou (editors), p. 51–64, November 2007.

[MTO+05] H. Mili, G. Tremblay, A. Obaid, R. B. Tamrout, E. Caillot, "Adding Semantics to Web Service Descriptions", *Rapport technique*, Université du Québec à Montréal / LATECE, February 2005.

[TC05] G. Tremblay, J. Chae, "Toward Specifying Contracts and Protocols for Web Services", *in: Proceedings of the 1st Montréal Conference on e-Technologies MCeTech'2005 (Montréal, Canada)*, R. Dssouli, H. Mili (editors), IEEE Computer Society Press, p. 73–85, January 2005.

5.2.2 Compilation of LOTOS NT

Participants: David Champelovier, Xavier Clerc, Hubert Garavel, Yves Guerte, Frédéric Lang, Wendelin Serwe, Jan Stoecker.

As regards the LOTOS NT language — a variant of E-LOTOS elaborated by the VASY project-team — we worked along three directions:

- We continued enhancing our TRAIAN compiler (see § 4.2), which generates C code from LOTOS NT data type and function definitions. TRAIAN is distributed on the Internet (see § 8.1) and used intensively within the VASY project-team as a development tool for compiler construction [3].

  In 2008, we corrected five bugs and enhanced TRAIAN in several respects: we improved the efficiency of the C code generated by TRAIAN when this code is compiled with optimizing GCC; we modified the generated C code so as to remove all warnings emitted by SUN's LINT code checker and many warnings emitted by INTEL's ICC compiler; we developed configuration files to support the LOTOS NT language in the EMACS, XEMACS, and JEDIT editors; we enhanced the testing environment for TRAIAN and enriched its non-regression testing database with new examples. A new version 2.6 of TRAIAN was released on February 27, 2008.

- In the context of the MULTIVAL contract (see § 6.2), we continued improving the LNT2LOTOS tool suite that translates (a large subset of) LOTOS NT into LOTOS, thus allowing the use of CADP to verify LOTOS NT descriptions. This tool suite is being used by BULL to model and verify critical parts of its FAME2 multiprocessor architecture (see § 5.3.1); it was also used to verify protocol specifications provided by AIRBUS (see § 5.3.5).

  In 2008, the LNT2LOTOS tool suite was enhanced significantly. Eight successive versions were released and the tool suite was ported to five 32-bit architectures and three 64-bit architectures.

  As regards the data type part of LOTOS NT, we added support for character and string constants, simplified the use of predefined operations in module declarations, and allowed end-users to specify the names of the LOTOS and C types and functions to be generated by the LNT2LOTOS translator.

  As regards the process part of LOTOS NT, we finalized the concrete syntax and static semantics of processes, and specified an algorithm translating a set of LOTOS NT processes into a collection of LOTOS types, functions, and processes. This algorithm was implemented almost entirely in LNT2LOTOS and is currently under validation.

  A new tool named LNT_CHECK was introduced to check the exhaustivity of "**case**" patterns occuring both in functions and processes.

  LNT2LOTOS is developed using the SYNTAX/TRAIAN technology. Currently, it represents 32,500 lines of code (3,900 lines of SYNTAX code, 26,100 lines of LOTOS NT code, and 2,500 lines of C code). The LOTOS NT reference manual was updated and grew from 61 pages (at the end of 2007) to 71 pages. Five demo examples illustrating the usage of the LNT2LOTOS tool suite, together with a non-regression database containing 300 LOTOS NT programs, were developed.

- As regards the compilation of temporal extensions, we continued the definition of a new intermediate form, named ATLANTIF, into which languages combining data, concurrency, and real-time (such as E-LOTOS and LOTOS NT) could be translated efficiently. ATLANTIF is based upon the NTIF model [5] for sequential processes with data, which we extended in several ways to support concurrency and real-time.

  In 2008, we defined the syntax and formal timed semantics of ATLANTIF, which are conservative extensions of the syntax and formal semantics of NTIF, and we showed that the timed semantics of ATLANTIF has the expected properties of *time additivity* (a sequence of delays is equivalent to a delay of their sum), *time determinism* (a given delay leads to a unique state), and *maximal progress of urgent actions* (no delay is allowed if an urgent action is possible).

  We also implemented a translator tool that takes as input the description of a system in ATLANTIF and generates either a network of timed automata (for verification using UPPAAL) or a time Petri net (for verification using TINA). A paper on ATLANTIF was accepted for publication in an international conference [38].

5.2.3 Source-Level Translations between Concurrent Languages

Participants: Xavier Clerc, Hubert Garavel, Claude Helmstetter, Rémi Hérilier, Frédéric Lang, Olivier Ponsini, Wendelin Serwe, Damien Thivolle.

Although process algebras are, from a technical point of view, the best formalism to describe concurrent systems, they are not used as widely as they could be [25]. Besides the steep learning curve of process algebras, which is traditionally mentioned as the main reason for this situation, it seems also that the process algebra community scattered its efforts by developing too many languages, similar in concept but incompatible in practice. Even the advent of two international standards, such as LOTOS (in 1989) and E-LOTOS (in 2001), did not remedy this fragmentation. To address this problem, we started investigating source-level translators from various process algebras into LOTOS, so as to widen the applicability of the CADP tools.

In 2008, besides the LNT2LOTOS tool suite (see § 5.2.2), we worked on the following translators:

- In the framework of the OPENEMBEDD (see § 6.3) and TOPCASED (see § 6.4) projects, and in cooperation with the LAAS-CNRS and IRIT laboratories, we continued the development of FIACRE (*Format Intermédiaire pour les Architectures de Composants Répartis Embarqués*). Derived from NTIF [5] and V-COTRE [BRV+03], FIACRE will be used as a pivot formalism between modeling languages (such as AADL, UML, or SYSML) and verification tools (such as CADP and TINA). FIACRE may also serve as bridge between synchronous languages and asynchronous systems, as illustrated by the translation from

[BRV+03] B. BERTHOMIEU, P. RIBET, F. VERNADAT, J. BERNARTT, J.-M. FARINES, J.-P. BODEVEIX, M. FILALI, G. PADIOU, P. MICHEL, P. FARAIL, P. GAUFILLET, P. DISSAUX, J.-L. LAMBERT, "Towards the verification of real-time systems in avionics: the COTRE approach", *in: Proceedings of the 8th International Workshop on Formal Methods for Industrial Critical Systems FMICS'2003 (Trondheim, Norway)*, T. Arts, W. Fokkink (editors), *Electronic Notes on Theoretical Computer Science*, *80*, Elsevier, p. 201–216, June 2003.

the SIGNAL synchronous language to FIACRE proposed in [Per08].

In 2008, we finalized the formal definition of the semantics of FIACRE 2.0 [39]. We continued developing the FLAC (*Fiacre to Lotos Adaptation Component*) tool, which allows the translation of the full FIACRE language (except priorities and time constraints) into LOTOS (4,700 lines of Standard ML code). FLAC takes as input a FIACRE specification and produces a behaviourally equivalent LOTOS specification, together with an auxiliary SVL script allowing to embed the original identifiers of the FIACRE specification into the transition labels of the graph generated from the LOTOS specification. We tested FLAC on 70 FIACRE examples and released it on the TOPCASED forge[2]. Papers on FIACRE and FLAC were published in an international conference [22] and in an ERCIM newsletter [44].

- We considered the process algebra FSP (*Finite State Processes*) defined in a popular textbook on concurrency [MK06] and supported by the LTSA tool designed at Imperial College (London, United Kingdom). For this language, we developed the FSP2LOTOS tool, which translates FSP into LOTOS, EXP.OPEN, and SVL code [SKLM07].

  In 2008, we continued the development of the FSP2LOTOS translator. Most notably, we extended FSP2LOTOS to support a larger subset of FSP: on our non-regression database of 1,040 examples, the percentage of FSP programs accepted by the translator grew from 71% to 88%. We also divided by two the average size of the LOTOS code generated by FSP2LOTOS and increased its readability. We enhanced the LOTOS encoding of FSP numbers and labels, leading to more efficient translations. Finally, numerous bugs were solved and a manual page was written for the translator.

- In the context of the INRIA/LETI collaboration (see § 7.1) and the MULTIVAL contract (see § 6.2), we continued the study of the process algebra CHP (*Communicating Hardware Processes*) for which the TIMA laboratory has developed a circuit synthesis tool named TAST [Ren05] and which is used by the LETI laboratory to describe the FAUST/MAGALI architecture (see § 5.3.2). Our prior work led to the development of the CHP2LOTOS tool, which translates CHP into LOTOS, our goal being to integrate formal verification tools into the design flow of complex asynchronous circuits.

  In 2008, we devised proof arguments to establish the correctness of CHP2LOTOS. Our overview paper on the CHP to LOTOS translation was accepted for publication in an international journal [19].

- In the context of the INRIA/LETI collaboration (see § 7.1) and the MULTIVAL contract (see § 6.2), we investigated the verification of TLM (*Transaction Level Model*)

[2] `http://gforge.enseeiht.fr/projects/fiacre-compil`

[Per08] J. C. PERALTA, "From Signal Kernel to Fiacre", *research report*, INRIA/ESPRESSO, 15 pages, 2008.

[MK06] J. MAGEE, J. KRAMER, *Concurrency: State Models and Java Programs*, edition 2006, Wiley, April 2006.

[SKLM07] G. SALAÜN, J. KRAMER, F. LANG, J. MAGEE, "Translating FSP into LOTOS and Networks of Automata", *in: Proceedings of the 6th International Conference on Integrated Formal Methods IFM'2007 (Oxford, United Kingdom)*, J. Davies, W. Schulte, J. S. Dong (editors), *Lecture Notes in Computer Science*, *4591*, Springer Verlag, p. 558–578, July 2007.

[Ren05] M. RENAUDIN, *TAST Compiler and TAST-CHP Language – Version 0.6*, TIMA Laboratory, CIS Group, 2005.

models. Compared to traditional RTL (*Register Transfer Level*) models, TLM allows faster simulation, simultaneous development of software and hardware, and earlier hardware/software partitioning. Among all languages supporting TLM, SYSTEMC [IEE05] emerges as an industrial standard. SYSTEMC is a C++ library providing both a high-level description language and a simulation kernel that involves a central (largely deterministic) scheduler ordering the actions of the different processes.

Our prior work led to two approaches for translating the PV (*Programmers View*) level of SYSTEMC/TLM (i.e., TLM models without explicit timing information) into LOTOS:

- Our first approach follows the "official" simulation semantics of SYSTEMC, keeping the scheduler as defined in [IEE05].
- Our second approach proposes a fully asynchronous semantics for SYSTEMC/TLM, getting rid of the central scheduler that is difficult to implement in parallel circuits and prevents certain execution sequences from being analyzed during formal verification.

In 2008, we pursued our research on this topic. Both our approaches (i.e., with and without the scheduler) were published in international conferences, namely [26] for the first approach and [36] for the second approach. A comparison of both approaches on the parameterized benchmark with n concurrent components given in [TCMM07] was done and reported in [26].

To improve the effectiveness of formal verification, we experimented various ways of coding SYSTEMC/TLM transactions in LOTOS (inlined, instantiated, or synchronized by rendezvous). We showed that inlining transactions in the initiator thread is, when possible, by far the most efficient encoding as regards the size of the state space generated with CADP (up to 15 times less states than using instantiated transactions).

In parallel, to ease the integration of SYSTEMC/TLM and LOTOS, we investigated how to reuse, within a LOTOS specification, existing fragments of code written in SYSTEMC. In a first step, we experimented this idea on the *Blitter Display* case-study provided to us by STMICROELECTRONICS (see § 5.3.4). In a second step, we undertook the design of a library that allows, from a LOTOS specification, to invoke SYSTEMC/TLM simulation steps.

5.3 Case Studies and Practical Applications

5.3.1 The FAME2 Architecture

Participants: Hubert Garavel, Holger Hermanns, Radu Mateescu, Meriem Zidouni.

In the context of the MULTIVAL (see § 6.2) contract, we studied together with BULL the MPI software layer and MPI benchmark applications to be run on FAME2 (*Flexible Architecture*

[IEE05] IEEE, "IEEE Standard SystemC Language Reference Manual", *IEEE Standard number 1666-2005*, Institution of Electrical and Electronic Engineers, December 2005.

[TCMM07] C. TRAULSEN, J. CORNET, M. MOY, F. MARANINCHI, "A SystemC/TLM semantics in Promela and its possible applications", *in: 14th International SPIN Workshop on Model Checking Software*, *Lecture Notes in Computer Science*, *4595*, Springer Verlag, p. 204–222, July 2007.

for Multiple Environments), a CC-NUMA multiprocessor architecture developed at BULL for teraflop mainframes and petaflop computing.

In 2008, our activities focused on the following aspects:

- We pursued the study of the MPI benchmark called "*ping-pong*" protocol, our goal being to predict the performance of this benchmark on FAME2 machines, in particular to estimate the latency of send/receive operations on different topologies, different software implementations of the MPI primitives, and different cache coherency protocols. The ping-pong benchmark consists of two parallel processes, which send to each other a data packet k times. The benchmark was specified using a combination of LOTOS code (to describe the behavior of processes) and C code (to describe the data structures of memory and caches). Several configurations were considered, by specifying two different implementations of the send/receive primitives (based on linked lists with locks and on lock-free buffers, respectively) and two cache coherency protocols (in which a variable written by a process becomes either owned by that process, or shared between that process and the previous owner, respectively).

  The performance analysis was carried out by extending the LOTOS specification with Markov delays and applying the BCG_MIN, DETERMINATOR, and BCG_STEADY tools of CADP in order to calculate the latency of the send/receive operations. The quality of the model was improved by decomposing the read/write accesses in two request/response phases, which reflects the real behaviour of the system accurately. Thus, the latencies predicted from the model were close (down to 6% of difference) to the experimental measures. The performance analysis also allowed to estimate the number of cache misses corresponding to each instruction, which indicates that the second send/receive protocol (based on lock-free buffers) and the first cache coherency protocol (in which a variable written by a process becomes owned by that process) provide the best performance among the configurations considered. An article describing this work was accepted for publication in an international conference [23].

- We undertook the study of the "*barrier*" primitive of MPI, which allows several parallel processes to synchronize (each process arriving at the barrier waits for all the others to arrive) before continuing their execution. The latency of the barrier primitive corresponds to the (average) time taken by a process to traverse the barrier, i.e., the time between the moment when it arrives and when it leaves the barrier. We specified, using the LOTOS and C languages, five protocols implementing the barrier primitive (*centralized*, *combining*, *tournament*, *dissemination*, and *tree-based*). These five protocols differ by the shared data structures and the synchronizations used. In addition to the classical read/write operations, all these protocols use a *fetch-and-decrement* operation, which consists of a read and a write operation executed atomically. The *combining* protocol involves non-tail recursion, which was eliminated by modeling an explicit stack.

  As regards functional verification, we identified five temporal properties characterizing the correct behaviour of several parallel processes traversing a barrier cyclically: deadlock freeness, correct access to private variables, absence of access to a null address in memory, mandatory traversal of the current barrier by all processes, and forbidden leaving of a barrier by a process before all the others arrived at the barrier. These properties were successfully verified on the five barrier protocols by using the EVALUATOR model checker

of CADP. The first three properties were expressed in regular alternation-free μ-calculus and checked using EVALUATOR 3.5, whereas the last two properties, which require to count the processes arriving at a barrier, were expressed in MCL and checked using EVALUATOR 4.0.

As regards performance evaluation, we extended the LOTOS specifications of the centralized and tree-based barrier protocols (involving cyclic parallel processes) with Markov delays. It turned out that the Markovian model contained nondeterminism caused by simultaneous accesses of concurrent processes to the same shared variable. This nondeterminism was eliminated by inserting very small Markov delays before the conflicting accesses. In order to fight state explosion, we generated the Markov chain semi-compositionally by using hand-crafted interfaces for the memory and cache processes, and by minimizing the intermediate graphs progressively modulo stochastic branching bisimulation using BCG_MIN, the whole process being automated using SVL scripts. This allowed to generate the final Markov chain for the centralized barrier with 6 processes and the tree-based barrier with 4 processes, and to compute the latencies of barrier traversals for these protocols using BCG_STEADY.

5.3.2 The FAUST/MAGALI Architectures

Participants: Radu Mateescu, Wendelin Serwe.

In the context of the INRIA/LETI collaboration (see § 7.1) and the MULTIVAL contract (see § 6.2), we pursued the study (started in 2005) of FAUST (*Flexible Architecture of Unified System for Telecom*), a platform based on NOC (*Network on Chip*) for wireless telecom applications (4G, MIMO, etc.), developed by the CEA/LETI laboratory [BCV+05]. Since 2007, we are focusing on the latest version, named MAGALI, of this architecture.

Together with LETI scientists (Francois Bertrand, Virang Shah, and Yvain Thonnart), we studied the communication interconnect, which routes packets (consisting of several 34-bit flits) between the 23 components of the circuit. At the block level, this interconnect is described in the hardware process calculus CHP (*Communicating Hardware Processes*) and implemented, at the RTL level, in asynchronous logic. The interconnect has 23 communication nodes, each of which consists of five input controllers and five output controllers. Each input controller dispatches incoming flits to one out of four output controllers, and each output controller arbitrates between four input controllers. Using the CHP2LOTOS compiler [19] and compositional generation, it was possible to produce the two graphs corresponding to the input controller (4.6 million states and 16 million transitions) and the output controller (5.5 million states and 36 million transitions).

In 2008, we helped LETI scientists to write μ-calculus formulas expressing the correctness of the output controller. By using the EVALUATOR 3.5 tool, it was possible to verify several properties, such as: routing of flits on a virtual channel does not change before the last flit of a packet (i.e., all flits of the same packet are routed similarly), an acknowledgment of routing

[BCV+05] E. BEIGNÉ, F. CLERMIDY, P. VIVET, A. CLOUARD, M. RENAUDIN, "An Asynchronous NoC Architecture Providing Low Latency Service and its Multi-Level Design Framework", *in: Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems ASYNC'05 (New York, USA)*, IEEE Computer Society Press, p. 54–63, March 2005.

is generated at the end of a packet, and routing a new packet happens only after an arbitration decision has been taken.

To verify the input controller, LETI scientists decided not to use model checking (as done in 2006 [SSTV07] for a prior version of the input controller), since the μ-calculus formulas would have been too complex (the reason for the complexity increase is the higher storage capacity per channel in the latest version of the input controller). Instead, they used equivalence checking and developed an abstract model of the input controller (300 lines of LOTOS code, about 4.6 million states) that was proven, using the BISIMULATOR tool, to be included (with respect to branching bisimulation) in the input controller, showing that the input controller contains all expected behaviors.

We also helped LETI scientists in their effort to build a co-simulation platform to compare the formal LOTOS model of MAGALI and the implementation of MAGALI given as a VHDL netlist. Using the EXEC/CÆSAR framework [11], it was possible to embed the LOTOS model into a SYSTEMC process that was executed (using the MODELSIM tool of MENTOR GRAPHICS) in combination with the netlist. In this approach, the LOTOS model randomly generates inputs that are sent as inputs to the netlist. For each output sent back by the netlist, one checks whether a corresponding output also exists in the LOTOS model. As a first result, this co-simulation platform revealed that the LOTOS model was generating spurious inputs that would never occur actually. LETI scientists solved this issue by adding constraints in the LOTOS model to generate only valid inputs (e.g., in a sequence of flits, an end-of-packet may only occur after a begin-of-packet).

5.3.3 The xSTream Architecture

Participants: Nicolas Coste, Hubert Garavel, Holger Hermanns, Etienne Lantreibecq, Wendelin Serwe.

In the context of the MULTIVAL contract (see § 6.2) together with STMICROELECTRONICS, we studied XSTREAM, a multiprocessor dataflow architecture for high performance embedded multimedia streaming applications. In this architecture, computation nodes (e.g., filters) communicate using XSTREAM queues connected by a NOC (*Network on Chip*). An XSTREAM queue generalizes a bounded FIFO queue in two ways: it provides additional primitives (such as *peek* to consult items in the middle of the queue, which is not possible with the standard *push*/*pop* primitives of FIFO queues), and a *backlog* (extra memory) to allow the increase of the queue size when the queue overflows.

During the last year, the XSTREAM queues have been studied extensively. In 2008, we focused on the NOC itself and its interaction with XSTREAM queues.

The XSTREAM NOC is composed of routers connected by direct communication links. First experiments showed that representing XSTREAM primitives in the routers by two events (request/response), as used for modeling XSTREAM queues, would lead to state space explosion.

[SSTV07] G. SALAÜN, W. SERWE, Y. THONNART, P. VIVET, "Formal Verification of CHP Specifications with CADP — Illustration on an Asynchronous Network-on-Chip", *in: Proceedings of the 13th IEEE International Symposium on Asynchronous Circuits and Systems ASYNC 2007 (Berkeley, California, USA)*, P. Beerel, M. Roncken, M. Greenstreet, M. Singh (editors), IEEE Computer Society Press, p. 73–82, March 2007.

Therefore, we decided to represent each xSTream primitive by a single Lotos event in a router. To reuse the Lotos models already developed for xSTream queues (in which each xSTream primitive is represented using two Lotos events), we introduced additional Lotos processes, called *network interfaces*, that convert one representation into the other.

We developed two successive models of an xSTream router. The first model (150 lines of Lotos code) uses a single Lotos process with eight parameters representing the status of each of the eight ports of the router. State space generation for this model was slow due to the number of conditions evaluating to false when computing the successor states. Therefore, we produced a second model (120 lines of Lotos code), in which each port is represented by a separate process, which reduced the generation time by a factor 50, still yielding the same state space.

In a first step, we considered a NoC composed of four routers, where each router is directly connected to all the other routers. For this model, we could generate the corresponding state space (about $5,600$ states and $20,000$ transitions) compositionally, the largest intermediate graph having one million states and 5 million transitions. By using the Bisimulator tool, we could show that this model (i) is not branching bisimilar to, but only contains (modulo the branching preorder) the parallel composition of two xSTream queues (after abstracting away the network interfaces and the routers); (ii) is branching bisimilar to the composition of two xSTream queues connected by two network interfaces (after abstracting away only the routers). By using the Evaluator tool, we could verify temporal logic formulas expressing that the additional behavior is correct.

In a second step, we considered the more intricate case of a NoC composed of six routers, which are no longer fully connected; this implies that a packet might traverse more than two routers, which in turn increases the number of messages that might be received by a router. This resulted in an increase of the state space size for each single router: a router in a NoC with four routers has less than 400 states and $140,000$ transitions, whereas a router in a NoC with six routers has up to 6.7 million states and 49 million transitions, which was too large for further composition with the other routers, network interfaces, and xSTream queues. Therefore, we abstracted away the differences between all "disturbing" messages (i.e., messages that do not directly concern the considered pair of xSTream queues), ignoring destination and routing information in all these messages. This allowed to simplify the model of a router by removing those ports that are only used by the abstracted messages. The state space corresponding to this simplified model of a router was reduced down to 240 states and $1,000$ transitions. An additional benefit was that the same router could be used for all routers of the NoC (due to an asymmetry in the architecture, in the first model all routers had different state spaces). By using the Bisimulator tool, we found that the parallel composition of two xSTream queues connected by network interfaces and a NoC with six abstract routers was branching bisimilar to the parallel composition of two xSTream queues connected by two network interfaces. However, we could not establish an equivalence relation between the two models of a router, as the simplified model of a router is no longer deterministic (abstracting away the routing information from disturbing messages introduces nondeterminism, since some disturbing messages can be dropped while others have to be forwarded to the next router). We therefore refined the abstraction, distinguishing two kinds of disturbing packets, those to be dropped and those to be forwarded. Comparing this refined model with the complete one, we still did not observe branching equivalence, but only a mutual inclusion (i.e., two simulations, but not a bisimulation).

We also continued performance evaluation studies to predict latency and throughput of communication between xSTREAM queues. For us, a key challenge is to combine probabilistic/stochastic information (e.g., the rates at which xSTREAM applications push and pop elements in and out of the queues) with precise timing information (e.g., memory access time). After exploring different techniques and tools, we devised an approach, called IPC (*Interactive Probabilistic Chains*), inspired by Interactive Markov Chains [Her02], but using probabilities instead of stochastic distributions and a central clock governing all delays. We defined a structured operational semantics for standard process algebra operators (sequential and parallel composition, nondeterministic choice, etc.) applied to IPC and proved that probabilistic branching bisimulation is a congruence for the parallel composition of IPC. Taking advantage of the open architecture of CADP, we prototyped a tool chain (6,400 lines of C code and 400 lines of PERL script), which we experimented on several examples.

5.3.4 The Blitter Display

Participants: Hubert Garavel, Claude Helmstetter, Olivier Ponsini, Wendelin Serwe.

In the context of the MULTIVAL contract (see § 6.2), we started to study whether the CADP tools could enrich with formal verification capabilities the current design flow of STMICROELECTRONICS, which is based on SYSTEMC/TLM. To this aim, STMICROELECTRONICS provided us with the *Blitter Display* (BDISP for short), a 2D-graphics co-processor implementing BLIT (*Block Image Transfer*) and numerous graphical operators, e.g., rotations, alpha blending, or Blue Ray disc decoding. The BDISP is software-controlled through instructions written in nodes of up to four application queues and two real-time composition queues. It is described by more than 25,000 lines of SYSTEMC/TLM code.

First, to palliate the absence of the STMICROELECTRONICS in-house development environment, we designed a minimal testbench, and were able to recompile the BDISP and pass all of the required tests within our own build environment. We then seeked to obtain a LOTOS model of the BDISP. This was not immediate, because the SYSTEMC/TLM model of the BDISP is mostly sequential and data-intensive, and because it is not primarily tailored to formal verification. For this reason, we focused on the queue management part, which is responsible for triggering and ordering the execution of the queues according to their priorities. We isolated the queue management part from the rest (graphical data treatment) and explicited the asynchronous concurrency that actually exists in the transactions between the BDISP and its environment.

The translation from SYSTEMC/TLM to LOTOS was done manually, but according to systematic rules. The SYSTEMC/TLM code modeling input/output communications and concurrency was translated to LOTOS, while the rest of the code (i.e., plain C++ code) was kept unmodified and invoked from the LOTOS model. Concretely:

- We translated the communication and concurrency primitives from SYSTEMC/TLM into LOTOS using our schedulerless translation rules (see § 5.2.3 and [36]). This produced 920 lines of LOTOS code.

[Her02] H. HERMANNS, *Interactive Markov Chains and the Quest for Quantified Quality*, *LNCS*, *2428*, Springer Verlag, 2002.

- We identified all variables composing the internal BDISP state and optimized their memory size on the state vector (e.g., replacement of integer variables by bit-fields of minimal length).
- We splitted the rest of the SYSTEMC code into functions testing the BDISP state and into functions modifying the BDISP state (5,550 lines of C++ code).
- We abstracted the queue modeling (e.g., compact representation of instruction nodes, memory addresses abstracted to an enumerated type, etc.).
- We developed an interface between the LOTOS and C++ codes (2,250 lines of C code).

This novel, hybrid approach reuses large parts of the original SYSTEMC/TLM model (so that the model under verification remains close to the original one) and leads to good performance using the CADP tools. After assembling all these code fragments together, we obtained an executable LOTOS/C++ model of the BDISP. We used the OCIS tool of CADP to perform step-by-step simulation and the EVALUATOR tool to prove correctness properties on several verification scenarios.

5.3.5 The Airbus TFTP Protocol

Participants: Xavier Clerc, Hubert Garavel, Damien Thivolle.

In the context of the TOPCASED project (see § 6.4), we started studying how CADP can be used to verify protocols written in SAM, a graphical synchronous language developed by AIRBUS.

Our first step was to equip SAM with a formal semantics that did not exist so far. With the help of Patrick Farail and Pierre Gaufillet (AIRBUS), we identified the SAM subset actually used at AIRBUS. We defined three simplifications of SAM diagrams (elimination of multi-ports, subsystems, and macro-states) and wrote a formal semantics document [43] for SAM, which was validated by AIRBUS and included in the SAM official documentation repository.

We then considered a case-study provided to us by AIRBUS, namely a ground/plane communication protocol based on TFTP (*Trivial File Transfer Protocol*), which operates above the standard UDP (*User Datagram Protocol*) layer. This protocol was specified as a SAM automaton having 7 states and 39 transitions.

To verify this protocol using CADP, we proposed a translation from SAM into LOTOS NT data types and functions. Basically, a SAM automaton is translated into a Mealy function that takes the current state and a list of input values, and produces the next state and a list of output values. Following this approach, we obtained a LOTOS NT version of the TFTP automaton.

Then, we modelled in LOTOS NT an entire system in which two synchronous TFTP automata execute asynchronously and communicate with each other using UDP links. We thus obtained a typical example of a GALS (*Globally Asynchronous, Locally Synchronous*) system. To express the unreliability of UDP, which may lose, duplicate and/or reorder messages, we designed various LOTOS NT models of UDP based on bounded FIFO queues and bag data structures. The LOTOS NT specification of the entire system was translated into LOTOS using the LNT2LOTOS translator (see § 5.2.2); this allowed to detect and correct several mistakes in the translator itself.

We then formulated 29 correctness properties in temporal logic and verified them on the generated LOTOS model using the EVALUATOR 3.5 and 4.0 model checkers of CADP. This allowed us to find 19 errors; many of these errors, even if not critical, were degrading the protocol performance (due to, e.g., useless packet retransmissions).

To quantify the impact of these errors, we used the EXECUTOR tool of CADP to simulate the LOTOS model. This allowed us to estimate, for each error, its individual impact on the performance.

Our results were presented at AIRBUS, and it was decided to intensify work in this direction, the goal being to have those techniques used internally by the company.

5.3.6 Other Case Studies

Service-oriented architectures are complex applications in which several business processes interact through WEB services. BPEL (*Business Process Execution Language*) [JE07] is a standardized language allowing to describe the behaviour and interaction of concrete WEB services and of abstract business processes. In collaboration with Sylvain Rampacek (Université de Bourgogne), we enhanced the WSMOD compiler from BPEL to discrete timed automata in order to generate graphs accepted as input by CADP. This new feature allowed to verify temporal properties involving discrete time (e.g., counting of clock ticks, timeout calculations, etc.) on BPEL specifications using the EVALUATOR 4.0 model checker. We experimented this modeling and verification approach for analyzing the behaviour of a WEB service for GPS navigation. This work led to a publication that won the best paper award in an international conference [31].

Semantic WEB applications have as underlying models directed graphs describing the resources involved and their properties. When these graphs become very large, as it happens for facility management descriptions as specified by the IFC (*Industrial Foundation Classes*) [ISO05] standard, their analysis can be efficiently carried out by using verification tools, such as CADP. In collaboration with Christophe Cruz, Christophe Nicolle, and Sylvain Rampacek (Université de Bourgogne), we developed a tool named RDF.OPEN for translating the RDF (*Resource Description Framework*) [KC04] semantic WEB language into the graph exploration API defined by OPEN/CÆSAR. This allows, on the one hand, to analyze RDF descriptions using the on the fly verification tools of CADP and, on the other hand, to convert RDF descriptions into BCG files, which can be subsequently queried by using either the XTL language or the graph manipulation primitives available in the BCG libraries. Additionally, a framework based on WEB services was developed for invoking the EVALUATOR, BISIMULATOR, and REDUCTOR tools of CADP remotely for analyzing graphs produced from RDF descriptions. These tools have been applied to a concrete case-study, namely to analyze large RDF graphs generated from facility management IFC files describing professional buildings.

[JE07] D. JORDAN, J. EVDEMON, "Web Services Business Process Execution Language Version 2.0", *Oasis standard*, OASIS, Billerica, Massachussets, April 2007.

[ISO05] ISO/PAS, "Industry Foundation Classes (IFC2x) Platform Specification", *International Standard number 16739:2005*, International Organization for Standardization — Automation Systems and Integration, 2005.

[KC04] G. KLYNE, J. J. CARROLL, "Resource Description Framework (RDF): Concepts and Abstract Syntax", *W3c recommendation*, W3C, February 2004.

Other teams also used the CADP toolbox for various case studies. To cite only recent work not already described in previous VASY activity reports, we can mention:

- the fault-based conformance testing of the SIP registrar [AWW08];
- the analysis of Franklin's algorithm for leader election in anonymous rings [BFPvdP08];
- the verification of asynchronous circuits [BS07];
- the analysis of first-class futures of distributed grid components [CHM08];
- the verification of distributed shared memory systems [CD06];
- the verification and adaptation of WF/.NET components [CSC+07,CSC+08];
- the design of a secure, verified, fair exchange DRM scheme [DNJ08];
- the verification of ERLANG telecommunication systems [GDH08];

[AWW08] B. K. AICHERNIG, M. WEIGLHOFER, F. WOTAWA, "Improving Fault-based Conformance Testing", *in: Proceedings of the 4th International Workshop on Model-Based Testing MBT'2008 (Budapest, Hungary)*, B. Finkbeiner, Y. Gurevich, A. K. Petrenko (editors), *Electronic Notes in Theoretical Computer Science*, *220*, Elsevier, p. 63–77, March 2008.

[BFPvdP08] R. BAKHSHI, W. FOKKINK, J. PANG, J. VAN DE POL, "Leader Election in Anonymous Rings: Franklin Goes Probabilistic", *in: Proceedings of the 5th IFIP International Conference on Theoretical Computer Science TCS'2008 (Milano, Italy)*, G. Ausiello, J. Karhumäki, G. Mauri, C.-H. L. Ong (editors), *IFIP*, *273*, Springer Verlag, p. 57–72, September 2008.

[BS07] M. BOUBEKEUR, P. P. SCHELLEKENS, "Automatic Optimization Techniques for Formal Verification of Asynchronous Circuits", *in: Proceedings of the 14th IEEE International Conference on Electronics, Circuits and Systems ICECS'2007 (Marrakech, Morocco)*, D. Bouami, E. M. Aboulhamid, M. Eleuldj, M. Zwolinski (editors), IEEE Computer Society Press, p. 283–286, December 2007.

[CHM08] A. CANSADO, L. HENRIO, E. MADELAINE, "Transparent First-Class Futures and Distributed Components", *in: Proceedings of the 5th International Workshop on Formal Aspects of Component Software FACS'2008 (Málaga, Spain)*, C. Canal, C. Pasareanu (editors), *Electronic Notes in Theoretical Computer Science*, Elsevier, September 2008.

[CD06] V. CHENNAREDDY, J. K. DEKA, "Formally Verifying the Distributed Shared Memory Weak Consistency Models", *in: Proceedings of the 14th International Conference on Advanced Computing and Communications ADCOM'2006 (Mangalore, India)*, G. S. Kumar, K. C. Sekaran, K. R. Venugopal, L. M. Patnaik, K. S. Trivedi (editors), IEEE Computer Society Press, p. 455–460, December 2006.

[CSC+07] J. CUBO, G. SALAÜN, C. CANAL, E. PIMENTEL, P. POIZAT, "Relating Model-Based Adaptation and Implementation Platforms: A Case Study with WF/.NET 3.0", *in: Proceedings of the 12th International Workshop on Component-Oriented Programming WCOP'2007 (Berlin, Germany)*, R. Reussner, C. Szyperski, W. Weck (editors), p. 9–13, July 2007.

[CSC+08] J. CUBO, G. SALAÜN, C. CANAL, E. PIMENTEL, P. POIZAT, "A Model-Based Approach to the Verification and Adaptation of WF/.NET Components", *in: Proceedings of the 4th International Workshop on Formal Aspects of Component Software FACS'2007 (Sophia-Antipolis)*, M. Lumpe, E. Madelaine (editors), *Electronic Notes in Theoretical Computer Science*, *215*, Elsevier, p. 39–55, June 2008.

[DNJ08] M. T. DASHTI, S. K. NAIR, H. JONKER, "Nuovo DRM Paradiso: Designing a Secure, Verified, Fair Exchange DRM Scheme", *Fundamenta Informaticae 89*, 4, 2008, p. 393–417.

[GDH08] Q. GUO, J. DERRICK, C. HOCH, "Verifying Erlang Telecommunication Systems with the Process Algebra μCRL", *in: Proceedings of the 28th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2008 (Tokyo, Japan)*, K. Suzuki, T. Higashino, K. Yasumoto, K. El-Fakih (editors), *Lecture Notes in Computer Science*, *5048*, Springer Verlag, p. 201–217, June 2008.

- the abstraction and analysis of clinical guidance trees [Tur08];
- the specification and verification of middlewares [RC07];
- the performance analysis of stimulus rich reactive interfaces [SBB08];
- the formal specification and verification of CORBA systems [Ros08].

Finally, a book chapter [20] describing the application of CADP to the specification and verification of a turntable system for drilling products was published.

6 Contracts and Grants with Industry

6.1 The EC-MOAN Project

Participants: Hubert Garavel, Radu Mateescu, Emilie Oudot, Anton Wijs.

VASY participates to the EC-MOAN (*Scalable modeling and analysis techniques to study emergent cell behavior: Understanding the E. coli stress response*) project no. 043235, funded by the FP6 NEST-PATH-COM European program. It gathers seven participants: INRIA Rhône-Alpes (VASY and IBIS project-teams), Université Joseph Fourier (Grenoble), University of Twente, Free University of Amsterdam, University of Edinburgh, CWI Amsterdam, and Masaryk University Brno. EC-MOAN aims at the development of new, scalable methods for modeling and analyzing integrated genetic, metabolic, and signaling networks, and the application of these methods for a better understanding of a bacterial model system.

EC-MOAN started on February 1st, 2007 for three years. In 2008, our efforts focused on defining and implementing the temporal logic CTRL [27], an extension of CTL [CES86] with regular expressions and fairness operators able to specify biologically-relevant properties (multistability, oscillations, etc.) in a concise and natural way [27]. We also contributed to the definition of temporal logic patterns, expressed in CTL and modal μ-calculus, which encode typical properties of biological interest [35, 34, 21, 40, 41]. Finally, we developed new algorithms that improve the performance of on the fly verification [28, 29, 42] and state space generation [33], in order to analyze efficiently the graphs resulting from qualitative simulation of

[Tur08] K. J. TURNER, "Abstraction and Analysis of Clinical Guidance Trees", *Journal of Biomedical Informatics*, 2008.

[RC07] N. S. ROSA, P. R. F. CUNHA, "A Formal Framework for Middleware Behavioural Specification", *ACM SIGSOFT Software Engineering Notes 32*, 2, 2007, p. 1–7.

[SBB08] L. SU, H. BOWMAN, P. BARNARD, "Performance of Reactive Interfaces in Stimulus Rich Environments, Applying Formal Methods and Cognitive Frameworks", *in : Proceedings of the 2nd International Workshop on Formal Methods for Interactive Systems FMIS'2007 (Lancaster, UK)*, A. Cerone, P. Curzon (editors), *Electronic Notes in Theoretical Computer Science*, *208*, Elsevier, p. 95–111, April 2008.

[Ros08] N. S. ROSA, "Behavioral Specification of Middleware Systems", *in : Process Algebra for Parallel and Distributed Processing*, M. Alexander and W. Gardner (editors), Chapman and Hall, 2008, ch. 6, p. 161–198.

[CES86] E. M. CLARKE, E. A. EMERSON, A. P. SISTLA, "Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications", *ACM Transactions on Programming Languages and Systems 8*, 2, April 1986, p. 244–263.

genetic regulatory networks, such as those produced by the GNA (*Genetic Network Analyzer*) tool developed by the IBIS project-team.

6.2 The Multival Project

Participants: David Champelovier, Nicolas Coste, Hubert Garavel, Yves Guerte, Rémi Hérilier, Holger Hermanns, Romain Lacroix, Frédéric Lang, Etienne Lantreibecq, Radu Mateescu, Louis Paternault, Olivier Ponsini, Wendelin Serwe, Meriem Zidouni.

MULTIVAL (*Validation of Multiprocessor Multithreaded Architectures*) is a project of MINALOGIC, the *pôle de compétitivité mondial* dedicated to micro-nano technologies and embedded software for systems on chip (EMSOC cluster of MINALOGIC). It is funded by the French government (*Fonds Unique Interministériel*) and *Conseil général de l'Isère*. MULTIVAL addresses verification and performance evaluation issues for three innovative asynchronous architectures developed by BULL, CEA/LETI, and STMICROELECTRONICS.

MULTIVAL started in December 2006 for three years. In 2008, we focused our activities on the enhancement of CADP (see § 5.1 and § 5.2) and case studies in collaboration with our partners to verify and predict the performance of the architectures FAME2 (see § 5.3.1), FAUST/MAGALI (see § 5.3.2), and XSTREAM (see § 5.3.3).

6.3 The OpenEmbedd Project

Participants: Xavier Clerc, Hubert Garavel, Frédéric Lang, Radu Mateescu, Wendelin Serwe, Jan Stoecker.

OPENEMBEDD is a French national project of ANR (*Agence Nationale de la Recherche*), initiated by RNTL (*Réseau National des Technologies Logicielles*). The goal of OPENEMBEDD is to develop an open-source, generic, standard software engineering platform for real-time embedded systems, such as those developed by AIRBUS, CS, FRANCE TELECOM, and THALES. Within an ECLIPSE framework, this platform will combine the principles of model-driven engineering with those of formal methods.

OPENEMBEDD started in May 2006 for three years. In 2008, we completed the definition of the FIACRE asynchronous intermediate model for embedded systems [39] and developed an automated translator from FIACRE to LOTOS (see § 5.2.3).

6.4 The Topcased Project

Participants: Hubert Garavel, Romain Lacroix, Frédéric Lang, Jeanne Merle, Sylvain Robert, Jan Stoecker, Damien Thivolle.

TOPCASED (*Toolkit in OPen-source for Critical Application and SystEms Development*) is a project of AESE, the French *pôle de compétitivité mondial* dedicated to aeronautics, space, and embedded systems. This project gathers 23 partners, including companies developing safety-critical systems such as AIRBUS (leader), ASTRIUM, ATOS ORIGIN, CS, SIEMENS VDO, and THALES AEROSPACE.

TOPCASED develops a modular, open-source, generic CASE (*Computer-Aided Software Engineering*) environment providing methods and tools for embedded system development, ranging from system and architecture specifications to software and hardware implementation through equipment definition. VASY contributes to the combination of model-driven engineering and formal methods for asynchronous systems.

TOPCASED started in August 2006 for 40 months. In 2008, we worked along the following lines:

- We proposed a formal semantics for the SAM graphical language developed by AIRBUS and also a translation from a subset of SAM to LOTOS NT, with the goal of verifying SAM specifications automatically. We applied this translation to obtain a formal model (a combination of LOTOS NT specifications and automata) of the TFTP protocol, which we subsequently analyzed using the CADP tools (see § 5.3.5).
- In collaboration with colleagues from LAAS-CNRS and IRIT (Toulouse, France), we completed the definition of FIACRE, an intermediate model for embedded systems with asynchrony and quantitative time [39, 22]. We also completed the development (started in 2007) of an automated translator from FIACRE to LOTOS [44] (see § 5.2.3).
- We participated in the TOPCASED Quality Group, which defines the quality policy for TOPCASED (in particular, a set of mandatory requirements) and evaluates the TOPCASED development activities.

H. Garavel is the INRIA representative at the TOPCASED executive committee, for which he served as the secretary during the elaboration phase of the TOPCASED proposal.

7 Other Grants and Activities

7.1 National Collaborations

From 2004 to 2008, the VASY project-team played an active role in the joint research center between INRIA Rhône-Alpes and the LETI laboratory of CEA-Grenoble. In collaboration with LETI scientists (Edith Beigné, François Bertrand, Fabien Clermidy, Virang Shah, Yvain Thonnart, and Pascal Vivet), VASY developed software tools for the design of asynchronous circuits and architectures such as GALS (*Globally Asynchronous Locally Synchronous*), NOCs (*Networks on Chip*), and SOCs (*Systems on Chip*). In 2008, this collaboration was pursued as part of the MULTIVAL project (see § 6.2).

Additionally, we collaborated in 2008 with several INRIA project-teams:

- ATLAS (Nantes): collaboration in the framework of the OPENEMBEDD national action (Jean Bézivin and Frédéric Jouault);
- ALPAGE (Rocquencourt): enhancements to the SYNTAX V6 compiler generation software (Pierre Boullier, Philippe Deschamp, and Benoît Sagot);
- ESPRESSO (Rennes): collaboration in the framework of the TOPCASED and OPENEMBEDD national actions (Jean-Pierre Talpin and Julio Peralta);

- Ibis (Rhône-Alpes): applications of model checking to biological systems (Estelle Dumas, Hidde de Jong, Pedro Monteiro, and Michel Page).

Beyond Inria, we had sustained scientific relations with the following teams:

- Laas-Cnrs laboratory (Toulouse): collaboration in the framework of the OpenEmbedd and Topcased projects (Bernard Berthomieu and François Vernadat);
- Le2i laboratory (Dijon): applications of model checking to Web services and business processes (Sylvain Rampacek). From December 2006 to September 2008, R. Mateescu, E. Oudot, A. Wijs, and M. Zidouni were hosted by Université de Bourgogne.

7.2 International Collaborations

The Vasy project-team is member of the Fmics (*Formal Methods for Industrial Critical Systems*) working group of Ercim (see `http://www.inrialpes.fr/vasy/fmics`). From July 1999 to July 2001, H. Garavel chaired this working group. Since July 2002, he is member of the Fmics Board, in charge of dissemination actions.

H. Garavel is a member of Ifip (*International Federation for Information Processing*) Technical Committee 1 (*Foundations of Computer Science*) Working Group 1.8 on Concurrency Theory chaired by Luca Aceto.

In addition to our partners in aforementioned contractual collaborations, we had scientific relations in 2008 with several international universities and research centers, including:

- Imperial College (Jeff Kramer and Jeff Magee),
- Liama, China (Claude Helmstetter and Vania Joloboff),
- Polytechnic University of Bucharest (Valentin Cristea),
- Saarland University (Pepijn Crouzen, Holger Hermanns, Sven Johr, and Reza Pulungan),
- University of Málaga (Gwen Salaün),
- University of Malta (Gordon Pace and Sandro Spina), and
- University of Twente (Jaco van de Pol).

7.3 Visits and Invitations

In 2008, we had the following scientific exchanges:

- Pascal Poizat (Université d'Evry, France) visited us on April 14–18, 2008.
- The annual Vasy seminar was held in Monthieux on June 10–13, 2008. In addition to the Vasy project team, Valentin Cristea (Polytechnic University of Bucharest) and Sylvain Rampacek (Université de Bourgogne) attended this seminar.

8 Dissemination

8.1 Software Dissemination and Internet Visibility

The VASY project-team distributes two main software tools: the CADP toolbox (see § 4.1) and the TRAIAN compiler (see § 4.2). In 2008, the main facts are the following:

- We prepared and distributed 14 successive beta-versions (from 2007-c to 2007-p "Zurich") of CADP.
- The number of license contracts signed for CADP increased from 394 to 411.
- We were requested to grant CADP licenses for 587 different computers in the world.
- The distribution of the TRAIAN compiler continued and a new version 2.6 of Traian (see § 4.2) was released on February 27, 2008.
- The TRAIAN compiler was downloaded by 44 different sites.

The VASY WEB site (see `http://www.inrialpes.fr/vasy`) was regularly updated with scientific contents, announcements, publications, etc.

In September 2007, we opened the "CADP Forum" (see `http://www.inrialpes.fr/vasy/cadp/forum.html`) for discussions regarding the CADP toolbox. By the end of December 2008, this forum had 88 registered users and 408 messages exchanged.

8.2 Program Committees

In 2008, the members of VASY took on the following responsibilities:

- F. Lang was a program committee member of NEPTUNE'2008 (*Nice Environment with a Process and Tools Using Norms and Examples*), ENST Paris, France, April 8-9, 2008.
- R. Mateescu was a program committee member of PDMC'2008 (*7th International Workshop on Parallel and Distributed Methods in verifiCation*), Budapest, Hungary, March 29, 2008.
- R. Mateescu was a program committee member of FMICS'2008 (*13th International Workshop on Formal Methods for Industrial Critical Systems*), l'Aquila, Italy, September 15-16, 2008.
- F. Lang was a program committee member of ECSA'2008 (*2nd European Conference on Software Architecture*), Paphos, Cyprus, September 29-October 1st, 2008.
- R. Mateescu was a program committee member of ICSEA'2008 (*3rd International Conference on Software Engineering Advances*), Sliema, Malta, October 26–31, 2008.

8.3 Lectures and Invited Conferences

In 2008, we gave talks in several international conferences and workshops (see bibliography below). Additionally:

- E. Oudot participated to the 3rd semestrial Ec-Moan meeting held at Masaryk University Brno (Czech Republic) on January 14–15, 2008. She gave a talk entitled "*Local Resolution of Boolean Equation Systems and its Applications to the Analysis of State Spaces*" on January 14, 2008.

- R. Mateescu gave a talk entitled "*Génération distribuée d'espaces d'états de grande taille avec* Cadp" at the Rge seminar of the Gdr Asr group of Cnrs held in Belfort (France) on February 7, 2008.

- H. Garavel gave a lecture entitled "*Le model checking à l'INRIA*" at the meeting preceding the signature of the Cea–Inria collaboration agreement in Grenoble (France) on March 21, 2008.

- X. Clerc and F. Lang participated to the 4th semestrial OpenEmbedd meeting held at Inria Sophia-Antipolis on March 26–27, 2008. X. Clerc gave a talk entitled "*Vérification d'un programme* Fiacre *avec* Cadp".

- Vasy organized the 5th quarterly Multival meeting, held at Inria Rhône-Alpes (France) on April 3, 2008. N. Coste gave a talk entitled "*Quantitative Evaluation in Embedded System Design: Validation of Multiprocessor Multithreaded Architectures*". H. Garavel gave a talk entitled "*Avancement des tâches* Multival Sp2 *de juin 2007 à mars 2008*". E. Lantreibecq gave a talk entitled "*Modeling a Queue with* NoC *Interconnect*". O. Ponsini gave a talk entitled "*A Schedulerless Semantics of* Tlm *via Translation into* Lotos".

- M. Zidouni participated to the Model35 Symposium on Perspectives in Modeling and Performance Analysis of Computer Systems and Networks held at Inria Paris-Rocquencourt (France) on April 2–3, 2008. She gave a talk entitled "*Performance Evaluation of MPI Benchmarks on CC-DSM Multiprocessor Architectures*" on April 3, 2008.

- F. Lang participated to the Neptune conference held at Enst Paris on April 8-9, 2008.

- H. Garavel participated to the *IPA (Institute for Programming research and Algorithmics) Spring Days on Integrating Formal Methods* held at Rhenen (The Netherlands) on May 7–9, 2008. He gave an invited talk entitled "*Integrating Formal Methods within a Process Calculi Framework*" on May 8, 2008.

- O. Ponsini and D. Thivolle presented the Cadp toolbox during the poster and demonstration sessions of the 15th International Symposium on Formal Methods Fm'08 (Turku, Finland) on May 28, 2008.

- F. Lang participated to the workshop in honor of Pierre Lescanne's 60th birthday held at Inria Nancy on May 29, 2008.

- M. Zidouni participated to AEP9 (*9ème Atelier en Evaluation de Performances*) held in Aussois (France) on June 1–4, 2008. She gave a talk entitled "*Evaluation des performances de benchmarks MPI sur des architectures multiprocesseur de type CC-DSM*" on June 3, 2008.

- R. Mateescu participated to the RGE seminar of the GDR ASR group of CNRS held in Dijon (France) on June 5, 2008.

- O. Ponsini presented the poster of R. Mateescu and E. Oudot entitled "*Efficient On-the-Fly Equivalence Checking using Boolean Equation Systems*" during the poster session of the 6th ACM-IEEE International Conference on Formal Methods and Models for Codesign MEMOCODE'2008 (Anaheim, USA) on June 5, 2008.

- R. Mateescu and A. Wijs participated to the 4th semestrial EC-MOAN meeting held at the University of Twente (Enschede, The Netherlands) on June 17–18, 2008. R. Mateescu gave two talks entitled "*Computation Tree Regular Logic for Genetic Regulatory Networks*" and "*Improved On-the-Fly Equivalence Checking using Boolean Equation Systems*" on June 18, 2008. A. Wijs gave a talk entitled "*Hierarchical Adaptive State Space Caching*" on June 18, 2008.

- N. Coste, X. Clerc, H. Garavel, H. Hermanns, Y. Guerte, R. Hérilier, R. Lacroix, E. Lantreibecq, R. Mateescu, O. Ponsini, W. Serwe, and M. Zidouni participated to the 6th quarterly MULTIVAL meeting held at CEA/LETI (Grenoble, France) on June 20, 2008. N. Coste gave a talk entitled "*Preuve de congruence pour la bisimulation de branchement probabiliste*". H. Garavel gave a talk entitled "*Avancement de* CADP *64 bits*". R. Lacroix gave a talk entitled "*Porting* SYNTAX *to 64-bit Computers*". E. Lantreibecq gave a talk entitled "*Modélisation d'un* NOC *à six noeuds*". R. Mateescu gave a talk entitled "*A Model Checking Language for Concurrent Value-Passing Systems*". O. Ponsini gave a talk entitled "*An Introduction to the Blitter Case-Study*". M. Zidouni gave a talk entitled "*Evaluation des performances* MPI *: cas du benchmark Barrier*".

- H. Garavel, O. Ponsini, and W. Serwe participated to a technical meeting held at STMICROELECTRONICS (Grenoble, France) to present the results of the "*Blitter*" case study (see § 5.3.4) on September 15, 2008. O. Ponsini gave a talk entitled "*Un modèle* LOTOS *du Blitter*".

- R. Mateescu participated to the *MPI (Max Planck Institute) Summer School on Verification Technology, Systems and Applications* held at Saarbrücken (Germany) on September 15–19, 2008. He gave an invited lecture entitled "*Model Checking of Action-Based Concurrent Systems*" on September 15–16, 2008.

- X. Clerc, N. Coste, H. Garavel, H. Hermanns, E. Lantreibecq, W. Serwe, and M. Zidouni participated to the 7th quarterly MULTIVAL meeting held at STMICROELECTRONICS (Paris, France) on September 26, 2008. N. Coste gave a talk entitled "*Flot d'évaluation de performance de* XSTREAM". X. Clerc gave a talk entitled "LNT2LOTOS*: Language and Translation into* LOTOS". H. Garavel gave a talk entitled "*Avancement du portage de* CADP *sur les architectures 64 bits*". H. Hermanns gave a talk entitled "*Probabilistic Counterexample-Guided Abstraction Refinement*". E. Lantreibecq gave a talk entitled "*Modélisation d'un* NOC *à six noeuds*". M. Zidouni gave a talk entitled "*Evaluation des performances* MPI *: cas du benchmark Barrier*".

- D. Thivolle gave a talk entitled "*Validation and Simulation of* TopCased Sam *Specifications using* Cadp" at Airbus (Blagnac, France) on October 3, 2008.
- H. Garavel, F. Lang, and J. Stoecker participated to the 5th semestrial OpenEmbedd meeting held at Inria Rocquencourt on November 13–14, 2008. H. Garavel gave a talk entitled "*Validation and Performance Evaluation of* Sildex/Sam *Specifications using* Cadp". J. Stoecker gave a talk entitled "*Parallel Processes with Real-Time and Data: the* Atlantif *Intermediate Format*".
- F. Lang gave an invited talk entitled "*Recent Developments and Improvements of the* Cadp *Toolbox*" at the Safa workshop (*Workshop of the Sophia-Antipolis Formal Analysis Group*) held at Inria Sophia-Antipolis on December 3, 2008.
- H. Garavel gave a lecture entitled "*Quelques réflexions sur la validation de systèmes embarqués*" before the Minalogic/EmSoC prospective committee (Grenoble, France) on December 16, 2008.
- N. Coste, H. Garavel, Y. Guerte, C. Helmstetter, R. Hérilier, E. Lantreibecq, R. Mateescu, L. Paternault, W. Serwe and M. Zidouni participated to the 8th quarterly Multival meeting held at STMicroelectronics (Grenoble, France) on December 18, 2008. N. Coste gave a talk entitled "*Avancement du flot d'évaluation de performance*". H. Garavel gave a talk entitled "*Avancement du portage des outils* Cadp *sur les architectures 64 bits*". M. Zidouni gave a talk entitled "*Modélisation et évaluation de performance des barrières* Mpi".

8.4 Teaching Activities

The Vasy project-team is a host team for the computer science master entitled "*Mathématiques, Informatique, spécialité : Systèmes et Logiciels*", common to Grenoble Inp and Université Joseph Fourier.

In 2008:

- H. Garavel, F. Lang, and W. Serwe gave, jointly with Pascal Raymond (Cnrs, Verimag), a course on "*Méthodes formelles de développement*" to the computer science engineering students of Cnam (*Conservatoire National des Arts et Métiers*) Grenoble (21 hours).
- F. Lang and W. Serwe gave the course on "*Temps réel*" to the 3rd year students of Ensimag (18 hours).
- R. Mateescu, together with Sylvain Rampacek, Arnaud da Costa, and Nader Embarek (Université de Bourgogne) gave the course on "*Méthodes formelles*" to the 5th year students of Esirem (86 hours).
- F. Lang was a reviewer for Edoardo Bonta's PhD thesis entitled "*Automatic Code Generation: from Process Algebraic Architectural Descriptions to Multithreaded Java Programs*", defended at the University of Bologna (Italy), March 2008.
- H. Garavel was a jury member of Hans Svensson's PhD thesis entitled "*Verification of Distributed Erlang Programs using Testing, Model Checking and Theorem Proving*", defended at Chalmers University of Technology (Gothenburg, Sweden) on April 21, 2008.

- F. Lang was a jury member of Bilal Kanso's MSc thesis entitled "*Utilisation de graphes d'attaques pour tester la sécurité de systèmes répartis*", defended at Université Joseph Fourier (Grenoble) on June 23, 2008.

- F. Lang was a jury member of Antonio Cansado's PhD thesis entitled "*Formal Specification and Verification of Distributed Component Systems*", defended at Université de Nice on December 4, 2008.

- R. Mateescu was a suppliant member of the "*commission de spécialistes*" at Université de Bourgogne (section 27).

8.5 Miscellaneous Activities

Within the MINALOGIC *pôle de compétitivité mondial*, H. Garavel is a member of the operational committee of the EMSOC cluster (*Embedded System on Chip*).

H. Garavel is a member of the scientific council of the GIS (*Groupement d'Intérêt Scientifique*) consortium 3SGS on supervision, safety, and security of large systems.

F. Lang participates to the "*commission du développement technologique*" in charge of selecting projects of INRIA Grenoble Rhône-Alpes.

O. Ponsini was the VASY representative within the working group for creating the WEB server of the LIG laboratory.

9 Bibliography

Reference Publications by the Team

[1] H. GARAVEL, H. HERMANNS, "On Combining Functional Verification and Performance Evaluation using CADP", *in : Proceedings of the 11th International Symposium of Formal Methods Europe FME'2002 (Copenhagen, Denmark)*, L.-H. Eriksson, P. A. Lindsay (editors), *Lecture Notes in Computer Science*, *2391*, Springer Verlag, p. 410–429, July 2002. Full version available as INRIA Research Report 4492.

[2] H. GARAVEL, F. LANG, R. MATEESCU, W. SERWE, "CADP 2006: A Toolbox for the Construction and Analysis of Distributed Processes", *in : Proceedings of the 19th International Conference on Computer Aided Verification CAV'2007 (Berlin, Germany)*, W. Damm, H. Hermanns (editors), *Lecture Notes in Computer Science*, *4590*, Springer Verlag, p. 158–163, July 2007.

[3] H. GARAVEL, F. LANG, R. MATEESCU, "Compiler Construction using LOTOS NT", *in : Proceedings of the 11th International Conference on Compiler Construction CC 2002 (Grenoble, France)*, N. Horspool (editor), *Lecture Notes in Computer Science*, *2304*, Springer Verlag, p. 9–13, April 2002.

[4] H. GARAVEL, F. LANG, "SVL: a Scripting Language for Compositional Verification", *in : Proceedings of the 21st IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2001 (Cheju Island, Korea)*, M. Kim, B. Chin, S. Kang, D. Lee (editors), IFIP, Kluwer Academic Publishers, p. 377–392, August 2001. Full version available as INRIA Research Report RR-4223.

[5] H. Garavel, F. Lang, "NTIF: A General Symbolic Model for Communicating Sequential Processes with Data", *in : Proceedings of the 22nd IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2002 (Houston, Texas, USA)*, D. Peled, M. Vardi (editors), *Lecture Notes in Computer Science*, *2529*, Springer Verlag, p. 276–291, November 2002. Full version available as Inria Research Report RR-4666.

[6] H. Garavel, R. Mateescu, I. Smarandache, "Parallel State Space Construction for Model-Checking", *in : Proceedings of the 8th International SPIN Workshop on Model Checking of Software SPIN'2001 (Toronto, Canada)*, M. B. Dwyer (editor), *Lecture Notes in Computer Science*, *2057*, Springer Verlag, p. 217–234, Berlin, May 2001. Full version available as Inria Research Report RR-4341.

[7] H. Garavel, W. Serwe, "State Space Reduction for Process Algebra Specifications", *Theoretical Computer Science 351*, 2, February 2006, p. 131–145.

[8] H. Garavel, J. Sifakis, "Compilation and Verification of LOTOS Specifications", *in : Proceedings of the 10th International Symposium on Protocol Specification, Testing and Verification (Ottawa, Canada)*, L. Logrippo, R. L. Probert, H. Ural (editors), IFIP, North-Holland, p. 379–394, June 1990.

[9] H. Garavel, M. Sighireanu, "Towards a Second Generation of Formal Description Techniques – Rationale for the Design of E-LOTOS", *in : Proceedings of the 3rd International Workshop on Formal Methods for Industrial Critical Systems FMICS'98 (Amsterdam, The Netherlands)*, J.-F. Groote, B. Luttik, J. van Wamel (editors), CWI, p. 187–230, Amsterdam, May 1998. Invited talk.

[10] H. Garavel, M. Sighireanu, "A Graphical Parallel Composition Operator for Process Algebras", *in : Proceedings of the Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, and Protocol Specification, Testing, and Verification FORTE/PSTV'99 (Beijing, China)*, J. Wu, Q. Gao, S. T. Chanson (editors), IFIP, Kluwer Academic Publishers, p. 185–202, October 1999.

[11] H. Garavel, C. Viho, M. Zendri, "System Design of a CC-NUMA Multiprocessor Architecture using Formal Specification, Model-Checking, Co-Simulation, and Test Generation", *Springer International Journal on Software Tools for Technology Transfer (STTT) 3*, 3, July 2001, p. 314–331, Full version available as Inria Research Report RR-4041.

[12] H. Garavel, "Compilation of LOTOS Abstract Data Types", *in : Proceedings of the 2nd International Conference on Formal Description Techniques FORTE'89 (Vancouver B.C., Canada)*, S. T. Vuong (editor), North-Holland, p. 147–162, December 1989.

[13] H. Garavel, "OPEN/CÆSAR: An Open Software Architecture for Verification, Simulation, and Testing", *in : Proceedings of the First International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'98 (Lisbon, Portugal)*, B. Steffen (editor), *Lecture Notes in Computer Science*, *1384*, Springer Verlag, p. 68–84, Berlin, March 1998. Full version available as Inria Research Report RR-3352.

[14] H. Garavel, "Défense et illustration des algèbres de processus", *in : Actes de l'Ecole d'été Temps Réel ETR 2003 (Toulouse, France)*, Z. Mammeri (editor), Institut de Recherche en Informatique de Toulouse, September 2003.

[15] R. Mateescu, M. Sighireanu, "Efficient On-the-Fly Model-Checking for Regular Alternation-Free Mu-Calculus", *Science of Computer Programming 46*, 3, March 2003, p. 255–281.

[16] R. Mateescu, "On-the-fly State Space Reductions for Weak Equivalences", *in : Proceedings of the 10th International Workshop on Formal Methods for Industrial Critical Systems FMICS'05 (Lisbon, Portugal)*, T. Margaria, M. Massink (editors), ERCIM, ACM Computer Society Press, p. 80–89, September 2005.

[17] R. Mateescu, "CAESAR_SOLVE: A Generic Library for On-the-Fly Resolution of Alternation-Free Boolean Equation Systems", *Springer International Journal on Software Tools for Technology Transfer (STTT) 8*, 1, February 2006, p. 37–56, Full version available as Inria Research Report RR-5948, July 2006.

Journal Articles and Book Chapters

[18] W. Fokkink, J. Pang, A. Wijs, "Is Timed Branching Bisimilarity a Congruence Indeed?", *Fundamenta Informaticae 87*, 3–4, 2008, p. 287–311.

[19] H. Garavel, G. Salaün, W. Serwe, "On the semantics of communicating hardware processes and their translation into LOTOS for the verification of asynchronous circuits with CADP", *Science of Computer Programming*, 2009, to appear.

[20] R. Mateescu, "Specification and Analysis of Asynchronous Systems using CADP", *in : Modeling and Verification of Real-Time Systems — Formalisms and Software Tools*, S. Merz and N. Navet (editors), ISTE publishing / John Wiley, 2008, ch. 5, p. 141–170.

[21] P. T. Monteiro, D. Ropers, R. Mateescu, A. T. Freitas, H. de Jong, "Temporal Logic Patterns for Querying Dynamic Models of Cellular Interaction Networks", *Bioinformatics 24*, 16, 2008, p. i227–i233.

Publications in Conferences and Workshops

[22] B. Berthomieu, J.-P. Bodeveix, P. Farail, M. Filali, H. Garavel, P. Gaufillet, F. Lang, F. Vernadat, "FIACRE: an Intermediate Language for Model Verification in the TOPCASED Environment", *in : Proceedings of the 4th European Congress on Embedded Real-Time Software ERTS'08 (Toulouse, France)*, J.-C. Laprie (editor), SIA (the French Society of Automobile Engineers), AAAF (the French Society of Aeronautic and Aerospace), and SEE (the French Society for Electricity, Electronics, and Information & Communication Technologies), January 2008.

[23] G. Chehaibar, M. Zidouni, R. Mateescu, "Modeling Multiprocessor Cache Protocol Impact on MPI Performance", *in : Proceedings of the 2009 IEEE International Workshop on Quantitative Evaluation of Large-Scale Systems and Technologies QuEST'09 (Bradford, UK)*, IEEE Computer Society, May 2009. to appear.

[24] N. Coste, H. Garavel, H. Hermanns, R. Hersemeule, Y. Thonnart, M. Zidouni, "Quantitative Evaluation in Embedded System Design: Validation of Multiprocessor Multithreaded Architectures", *in : Special Session at Design, Automation & Test in Europe DATE'08 (Munich, Germany)*, March 2008.

[25] H. Garavel, "Reflections on the Future of Concurrency Theory in General and Process Calculi in Particular", *in : Proceedings of the LIX Colloquium on Emerging Trends in Concurrency Theory (Ecole Polytechnique de Paris, France), November 13–15, 2006*, C. Palamidessi, F. D. Valencia (editors), *Electronic Notes in Theoretical Computer Science*, *209*, Elsevier Science Publishers, p. 149–164, April 2008. Also available as INRIA Research Report RR-6368.

[26] C. Helmstetter, O. Ponsini, "A Comparison of Two SystemC/TLM Semantics for Formal Verification", *in : Proceedings of the 6th ACM-IEEE International Conference on Formal Methods and Models for Codesign MEMOCODE'2008 (Anaheim, CA, USA)*, IEEE Computer Society Press, p. 59–68, June 2008.

[27] R. Mateescu, P. T. Monteiro, E. Dumas, H. de Jong, "Computation Tree Regular Logic for Genetic Regulatory Networks", *in : Proceedings of the 6th International Symposium on Automated Technology for Verification and Analysis ATVA'08 (Seoul, South Korea)*, S. D. Cha, J.-Y. Choi, M. Kim, I. Lee, M. Viswanathan (editors), *Lecture Notes in Computer Science*, *5311*, Springer Verlag, p. 48–63, October 2008. Full version available as INRIA Research Report RR-6521.

[28] R. Mateescu, E. Oudot, "Bisimulator 2.0: An On-the-Fly Equivalence Checker based on Boolean Equation Systems", *in : Proceedings of the 6th ACM-IEEE International Conference on Formal Methods and Models for Codesign MEMOCODE'2008 (Anaheim, CA, USA)*, IEEE Computer Society Press, p. 73–74, June 2008.

[29] R. Mateescu, E. Oudot, "Improved On-the-Fly Equivalence Checking using Boolean Equation Systems", *in : Proceedings of the 15th International SPIN Workshop on Model Checking of Software SPIN'2008 (Los Angeles, USA)*, K. Havelund, R. Majumdar, J. Palberg (editors), *Lecture Notes in Computer Science*, 5156, Springer Verlag, p. 196–213, August 2008. Full version available as INRIA Research Report RR-6777.

[30] R. Mateescu, P. Poizat, G. Salaün, "Adaptation of Service Protocols using Process Algebra and On-the-Fly Reduction Techniques", *in : Proceedings of the 6th International Conference on Service Oriented Computing ICSOC'08 (Sydney, Australia)*, A. Bouguettaya, I. Krueger, T. Margaria (editors), *Lecture Notes in Computer Science*, *5364*, Springer Verlag, p. 84–99, December 2008.

[31] R. Mateescu, S. Rampacek, "Formal Modeling and Discrete-Time Analysis of BPEL Web Services", *in : Proceedings of the 4th International Workshop on Enterprise and Organizational Modeling and Simulation EOMAS'08 (Montpellier, France)*, J. Barjis, M. M. Narasipuram, P. Rittgen (editors), *Lecture Notes in Business Information Processing*, *10*, Springer Verlag, p. 179–193, June 2008.

[32] R. Mateescu, D. Thivolle, "A Model Checking Language for Concurrent Value-Passing Systems", *in : Proceedings of the 15th International Symposium on Formal Methods FM'08 (Turku, Finland)*, J. Cuellar, T. Maibaum, K. Sere (editors), *Lecture Notes in Computer Science*, 5014, Springer Verlag, p. 148–164, May 2008.

[33] R. Mateescu, A. Wijs, "Hierarchical Adaptive State Space Caching based on Level Sampling", *in : Proceedings of the 15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'09 (York, UK)*, S. Kowalewski, A. Philippou (editors), *Lecture Notes in Computer Science*, Springer Verlag, March 2009. to appear.

[34] P. T. Monteiro, D. Ropers, R. Mateescu, A. T. Freitas, H. de Jong, "Temporal Logic Patterns for Querying Dynamic Models of Cellular Interaction Networks", *in : Proceedings of the 7th European Conference on Computational Biology ECCB'08 (Cagliari, Sardinia-Italy)*, A. Tramontano (editor), September 2008. Full version available as INRIA Research Report RR-6470.

[35] P. T. Monteiro, D. Ropers, R. Mateescu, A. T. Freitas, H. de Jong, "Temporal Logic Patterns for Querying Qualitative Models of Genetic Interaction Networks", *in : Proceedings of the 18th European Conference on Artificial Intelligence ECAI'08 (Patras, Greece)*, M. Ghallab (editor), IOS Press, p. 229–233, July 2008.

[36] O. Ponsini, W. Serwe, "A Schedulerless Semantics of TLM Models Written in SystemC via Translation into LOTOS", *in : Proceedings of the 15th International Symposium on Formal Methods FM'08 (Turku, Finland)*, J. Cuellar, T. Maibaum, K. Sere (editors), *Lecture Notes in Computer Science*, 5014, Springer Verlag, p. 278–293, May 2008.

[37] S. Spina, G. Pace, F. Lang, "Automatic Interface Generation for Compositional Verification", *in : Proceedings of the 5th Computer Science Annual Workshop CSAW'2007 (Msida, Malta)*, C. Borg, S. Spina, J. Abela (editors), University of Malta, p. 234–247, January 2008.

[38] J. Stoecker, F. Lang, H. Garavel, "Parallel Processes with Real-Time and Data: The ATLANTIF Intermediate Format", *in : Proceedings of the 7th International Conference on Integrated Formal Methods IFM'09 (Düsseldorf, Germany)*, M. Leuschel, H. Wehrheim (editors), *Lecture Notes in Computer Science*, Springer Verlag, February 2009. to appear.

Research Reports and Internal Publications

[39] B. Berthomieu, J.-P. Bodeveix, M. Filali, H. Garavel, F. Lang, F. Peres, R. Saad, J. Stoecker, F. Vernadat, "The Syntax and Semantics of Fiacre – version 2.0", *Project deliverable F3.2.2 (updated)*, AESE (*pôle de compétitivité mondial Midi-Pyrénées & Aquitaine: Aéronautique, Espace et Systèmes Embarqués*) project Topcased, April 2008.

[40] L. Brim, H. de Jong, R. Mateescu, "Definition and Temporal Logic Translation of Query Templates", *Project deliverable d.3.1*, FP6-NEST-STREP 043235 project EC-MOAN, March 2008.

[41] H. de Jong, R. Mateescu, H. Westerhoff, J. Geiselmann, I. Goryanin, "Properties of E. Coli Carbon and Nitrogen Metabolism", *Project deliverable d.4.1*, FP6-NEST-STREP 043235 project EC-MOAN, August 2008.

[42] R. Mateescu, H. de Jong, L. Brim, J. van de Pol, "Model Checking Algorithms for the Properties Encoding the Templates", *Project deliverable d.3.2*, FP6-NEST-STREP 043235 project EC-MOAN, August 2008.

[43] D. Thivolle, H. Garavel, X. Clerc, "Présentation du langage SAM d'Airbus", *research report*, INRIA/VASY, 16 pages, 2008.

Miscellaneous

[44] B. Berthomieu, H. Garavel, F. Lang, F. Vernadat, "Verifying Dynamic Properties of Industrial Critical Systems Using TOPCASED/FIACRE", ERCIM News vol. 75, p. 32–33, October 2008.

[45] H. de Jong, D. Ropers, R. Mateescu, J. Geiselmann, "Bioinformatique : de la cellule à la puce", La Recherche no. 419, May 2008.