INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team VASY

Validation of Systems

Rhône-Alpes

THEME COM

Table of contents

1. Team

Head of Team
Hubert Garavel [DR2 INRIA]

Administrative Assistant
Elodie Toihein

Inria Staff
Radu Mateescu [CR1 INRIA]
Frédéric Lang [CR1 INRIA]
Wendelin Serwe [CR2 INRIA]

Software Engineers
Damien Bergamini [until January 30, 2005]
David Champelovier

Post-Doctoral Fellow
Gwen Salaün

Ph. D. Student
Christophe Joubert

Student Interns
Jerôme Fereyre [CNAM Grenoble, since November 30, 2005]
Nathalie Lépy [CNAM Grenoble, since November 1st, 2005]
Abdul Malik Khan [Université Joseph Fourier (Grenoble), since November 1st, 2005]

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 from 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).

- Between 1992 and 2001, we contributed to the revision of LOTOS undertaken within ISO. This led to the definition of E-LOTOS (*Enhanced*-LOTOS, ISO/IEC standard 15437:2001), 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*) [11], [1], 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 for LOTOS NT a prototype compiler named TRAIAN (see § 4.2).

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

3.1. 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, bus arbitration protocols, cache coherency protocols, hardware/software codesign;
- *Databases:* transaction protocols, distributed knowledge bases, stock management;
- *Consumer electronics:* audiovisual remote control, video on-demand, FIREWIRE bus, home networking;
- *Security protocols:* authentication, electronic transactions, cryptographic key distribution;
- *Embedded systems:* smart-card applications, air traffic control;
- *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, etc.

4. Software

4.1. The CADP Toolbox

Participants: Damien Bergamini, David Champelovier, Hubert Garavel [contact person], Christophe Joubert, Frédéric Lang, Radu Mateescu, Wendelin Serwe.

We maintain and enhance CADP (*Construction and Analysis of Distributed Processes* – formerly known as CÆSAR/ALDÉBARAN *Development Package*), 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 [2] 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 [10] 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 [3] 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 from 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 in 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 on the fly,
  - DETERMINATOR, which eliminates 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 [48], CTL [44], ACTL [45], 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 team of INRIA Rennes.

The CADP tools are well-integrated and can be accessed easily using either the EUCALYPTUS graphical interface or the SVL [5] 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], Frédéric Lang.

We develop a compiler named TRAIAN for translating descriptions written in the LOTOS NT 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 [7]. The recent compilers developed by the VASY team — namely AAL (see § 6.1), CHP2LOTOS (see § 5.2.3), EVALUATOR 4.0, EXP.OPEN 2.0 (see § 5.1.5), NTIF (see § 5.2.2), and SVL (see § 5.1.5) — 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 CÆSAR_SOLVE Library

Participant: Radu Mateescu.

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 § 5.1.2), the model checker EVALUATOR 3.5 (see § 5.1.3), and the minimization tool REDUCTOR 4.0 (see § 5.1.4). 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 [3].

The CÆSAR_SOLVE library provides four different resolution algorithms: A1 and A2 are general algorithms based upon depth-first, respectively breadth-first, traversals of boolean graphs; A3 and A4 are optimized for the case of acyclic, respectively disjunctive/conjunctive, boolean graphs; they are based upon memory-efficient depth-first traversals of boolean graphs. All these algorithms can generate diagnostics explaining why a result is true or false (examples and counterexamples).

In 2005, the CÆSAR_SOLVE library ($11,600$ lines of C code) was extended and improved as follows:

- The library interface was enhanced with new types and functions to facilitate the definition of boolean equation systems. Also, a bug was corrected in the diagnostic generation mechanism of algorithm A2.
- A new resolution algorithm A5 was added to the library. This algorithm, based upon a depth-first search of the boolean graph, improves over algorithms A1–A4 by performing an early detection of examples (resp. counterexamples) in greatest (resp. least) fixed point equation blocks. This detection is based upon a generalization of Tarjan's algorithm for computing strongly connected components. Algorithm A5 proves to be much faster (between one and two orders of magnitude) than all the other algorithms of CÆSAR_SOLVE when it is invoked many times on the same equation block, e.g., for detecting τ-confluent or redundant transitions during the on the fly reductions performed by the REDUCTOR tool (see § 5.1.4).

A journal paper about the CÆSAR_SOLVE library was accepted for publication [21].

5.1.2. The BISIMULATOR Tool

Participants: David Champelovier, Radu Mateescu.

BISIMULATOR is an equivalence checker, which 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 § 5.1.1). 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 2005, we continued the development of the BISIMULATOR tool ($15,300$ lines of C code):

- The tool was enhanced with comparisons modulo the trace equivalence relation, the weak trace equivalence relation (which considers only visible transitions), and their associated preorder relations. The generation of counterexamples for these equivalences and their preorders was also implemented.
- The encoding of branching equivalence in terms of boolean equation systems was enhanced in order to reduce the number of τ-closures (transitive reflexive closures over τ-transitions) computed when one of the states being compared does not have outgoing τ-transitions. The new encoding allows to identify on the fly the cases when branching equivalence becomes identical to $\tau^*.a$ equivalence, and to simplify the equations accordingly. This can reduce the number of boolean variables by up to 40%.
- BISIMULATOR was coupled with the OCIS interactive simulator in order to allow a scenario contained in a BCG graph to be replayed interactively during the current OCIS simulation session. Typically, the BCG graph can be either a simulation scenario previously explored and saved using OCIS, or an execution trace produced by EXHIBITOR or EXECUTOR, or a diagnostic generated by EVALUATOR (see § 5.1.3) for a temporal logic property. BISIMULATOR allows OCIS to determine whether this BCG graph is a subset or not of the graph being explored during the current OCIS session, which amounts to checking graph inclusion modulo the preorder associated to strong equivalence. If so, a "positive" diagnostic (i.e., an example) is generated, which can be subsequently read and replayed by OCIS as an ordinary simulation scenario. This feature required the generation of "positive" diagnostics by BISIMULATOR, which so far only generated "negative" ones.

The BISIMULATOR tool led to a publication [23].

5.1.3. The EVALUATOR Tool

Participant: Radu Mateescu.

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 2005, we continued the development of the EVALUATOR tool. This led to a new version EVALUATOR 3.5 ($5,600$ lines of SYNTAX/FNC2 code and $5,100$ lines of C code) that supersedes the previous version 3.0, with the following enhancements:

- EVALUATOR 3.5 uses the resolution algorithms provided by the CÆSAR_SOLVE library (see § 5.1.1) whereas EVALUATOR 3.0 contained an ad hoc resolution engine. This improves modularity by clearly separating the translation of the verification problem into a boolean equation system (this is done in EVALUATOR) from the resolution itself (this is done in CÆSAR_SOLVE).
- The analysis of regular alternation-free μ-calculus formulas was enhanced with the detection of formulas that lead to disjunctive or conjunctive boolean equation systems. These systems can be solved more efficiently using algorithm A4 of CÆSAR_SOLVE, which does not keep in memory the dependencies between boolean variables. Since most of the formulas encountered in practice are of this type, this enhancement resulted in important memory reductions (proportional to the number of transitions in the graph being checked) with respect to EVALUATOR 3.0.
- Another optimization, performed on the system of modal equations used as intermediate representation by the tool, consisted in expanding on-line the propositional variables which occurred only once in the right-hand side of an equation. On most practical examples, this reduced by a factor of 3 the number of variables and induced the same reduction on the time and memory necessary for resolution.
- The generation of diagnostics (examples and counterexamples) was improved in order to reflect more accurately the structure of the temporal formulas. In the diagnostics produced by EVALUATOR 3.0, each state was associated to a state of the graph being checked, which caused the duplication of transitions in the diagnostic. For instance, when evaluating the formula "$\langle a.a.a \rangle\, true$" on the graph consisting of a single a-loop "$s \xrightarrow{a} s$", the diagnostic produced by EVALUATOR 3.0 was the graph with a single state s and three a-loop transitions attached to s. Instead, the diagnostic produced by EVALUATOR 3.5 is the sequence $s_1 \xrightarrow{a} s_2 \xrightarrow{a} s_3 \xrightarrow{a} s_4$, which is a better explanation that the formula requires to traverse three successive a-transitions.
- Several new command-line options were added to EVALUATOR 3.5 to benefit from all features of CÆSAR_SOLVE, namely: use of the breadth-first search based algorithm A2 to produce small-depth diagnostics, use of the memory-efficient algorithm A3 to check properties on acyclic graphs, and possibility to display the underlying boolean equation system in a textual form.

A detailed manual page for EVALUATOR 3.5 was written [36] and the tool became part of CADP in February 2005.

5.1.4. The REDUCTOR Tool

Participants: Frédéric Lang, Radu Mateescu.

The CADP toolbox contains a tool named REDUCTOR 3.0 that performs exhaustive reachability analysis combined with elimination of internal transitions on the fly (this preserves $\tau^*.a$ equivalence).

Also, the VASY team developed in 2003 a prototype tool [16] implementing τ-confluence reduction [47], a form of partial order reduction that preserves branching equivalence. This reduction is a mean to fight state explosion by trying to avoid the exploration of redundant interleavings resulting from independent τ-transitions. Indeed, experiments on various communication protocols and distributed systems have shown that τ-confluence may reduce the number of states and transitions by up to 3 orders of magnitude.

In 2005, both tools were merged into a single one, leading to version 4.0 of REDUCTOR. This new tool ($2,000$ lines of C code) operates on graphs represented implicitly using the OPEN/CÆSAR environment and provides six reduction algorithms:

- It can eliminate both τ-transitions and the so-called *redundant* transitions [56], still preserving safety equivalence.
- It can eliminate all τ-transitions, still preserving $\tau^*.a$ equivalence.
- It can eliminate all circuits of τ transitions, still preserving branching equivalence (this reduction is called τ-compression).
- It can perform τ-confluence reduction, still preserving branching equivalence.
- It can eliminate duplicate transitions, still preserving strong equivalence.
- It can fully minimize a graph modulo strong equivalence.

The 1st, 4th, and 6th reductions above are obtained by encoding the reduction problem into a boolean equation system that is resolved on the fly using algorithm A5 of the CÆSAR_SOLVE library (see § 5.1.1). The 6th reduction is "orthogonal" in the sense that it can be combined with any of the five other reductions.

A detailed manual page for REDUCTOR 4.0 was written [37] and the tool became part of CADP in October 2005. The τ-compression and τ-confluence reductions led to a publication [29].

5.1.5. Compositional Verification Tools

Participants: Frédéric Lang, Wendelin Serwe.

The CADP toolbox contains various tools dedicated to compositional verification, among which PROJECTOR 2.0, EXP.OPEN 2.0, and SVL play a central role.

PROJECTOR 2.0 is a tool (totally rewritten in 2002) that implements behaviour abstraction [46], [52] by taking into account interface constraints. In 2005, we improved its efficiency by introducing a hash function specifically adapted to state products. On real examples provided by the Technical University of Eindhoven, the execution time of PROJECTOR was divided by a factor of up to four.

EXP.OPEN 2.0 is a tool that explores on the fly the graph corresponding to a network of communicating automata (represented as a set of BCG files). These automata are composed together in parallel using either algebraic operators (as in CCS, CSP, LOTOS, and μCRL), "graphical" operators (as in E-LOTOS [50] and LOTOS NT), or synchronization vectors (as in the MEC and FC2 tools). Additional operators are available to hide and/or rename labels (using regular expressions) and to cut certain transitions. In 2005, we enhanced EXP.OPEN along the following lines:

- Following joint work with Jaco van de Pol (CWI, Amsterdam) in the framework of the SENVA collaboration (see § 7.2), we corrected two problems related to the support of μCRL in EXP.OPEN.
- We added options to obtain static information about the network of communicating automata, such as a list of the labels that potentially belong to the product and the size of each BCG graph in the network.
- We improved the algorithm for enumerating the successors of a given state, which reduced the generation time by $20\ \%$ on average, with a constant negligible memory overhead.
- We implemented two partial order reduction techniques, one preserving the deadlocks and the other one preserving the weak traces of the network of automata, thus extending the family of partial order reductions already available in EXP.OPEN.

EXP.OPEN was used in the framework of the FIACRE national action (see § 7.1) and we developed three new demo examples to illustrate the recent functionalities of EXP.OPEN (see § 5.3). An article about EXP.OPEN was published in an international conference [28].

SVL (*Script Verification Language*) is both a high level language for expressing complex verification scenarios and a compiler dedicated to this language. In 2005, we enhanced SVL along the following lines:

- We added support for two new equivalence relations, namely trace and weak trace equivalences, which can be used for graph comparison and reduction.
- We added a new operator called "*refined abstraction*", which allows to generate the graph of a process under constraints generated automatically using EXP.OPEN.
- We adapted SVL so that, depending on the equivalence to be preserved, it invokes EXP.OPEN with the most appropriate partial order reduction, which is inferred from the semantic context automatically.

5.1.6. Parallel and Distributed Verification Tools

Participants: David Champelovier, Hubert Garavel, Christophe Joubert, Radu Mateescu.

Enumerative verification algorithms need to explore and store very large graphs and, thus, are often limited by the capabilities of current sequential machines. To push forward the limits, we are studying parallel and distributed algorithms adapted to the clusters of PCs and networks of workstations available in most research laboratories.

As a first goal, we focused on parallelizing the graph construction algorithm, which is a bottleneck for verification, as it requires a considerable amount of memory to store all reachable states. For this purpose, we developed two tools [8]: DISTRIBUTOR splits the construction of a graph over N machines communicating using TCP/IP sockets; each machine builds a graph fragment, the distribution of states between the machines being determined by a static hash function; BCG_MERGE merges the N graph fragments constructed by DISTRIBUTOR to produce the entire graph.

In 2005, this first phase was completed. DISTRIBUTOR 3.0 and BCG_MERGE 3.0 became parts of CADP in January 2005 and a manual page for DISTRIBUTOR was written [35]. These tools were demonstrated at several occasions, including the PDMC'2005 international workshop (see § 8.2). A tool paper was accepted for publication [25].

As a second goal, we aim at parallelizing on the fly verification itself. Because the CÆSAR_SOLVE library (see § 5.1.1) is our central verification engine for both model checking, e.g., in the EVALUATOR tool (see § 5.1.3), and equivalence checking, e.g., in the BISIMULATOR (see § 5.1.2) and REDUCTOR tools (see § 5.1.4), we target at the development of a distributed version of the CÆSAR_SOLVE library that could solve boolean equation systems on the fly using several machines.

In 2005, this work progressed as follows:

- We continued the development of a distributed version of CÆSAR_SOLVE (currently $17,000$ lines of C code) [17]. The former distributed resolution algorithm [26], which could only handle boolean equation systems containing one single equation block, was enhanced to deal with multiple blocks. The enhanced algorithm combines a depth-first search traversal of the dependency graph between blocks (which is supposed to be acyclic) and a breadth-first search traversal of the boolean graphs associated to blocks, both performed in a distributed manner. For single block boolean equation systems, the enhanced algorithm exhibits almost the same performance as the former algorithm.
- The distributed algorithm for generating diagnostics on the fly was also enhanced to handle boolean equations systems with multiple blocks.
- The distributed algorithm for termination detection was improved to detect partial resolutions of the blocks, i.e., the fact that all boolean variables present in a region of a block have their final value computed. This allows to propagate the values of these variables along backward dependencies and, thus, to achieve a good distribution of the simultaneous resolution of all equation blocks.

- We implemented a prototype connection of the EVALUATOR 3.5 model checker to the distributed version of CÆSAR_SOLVE. Experiments were performed on the IDPOT cluster of PCs using various graphs taken from the VLTS benchmark suite and CADP demo examples. We checked properties ranging from basic deadlock and livelock detection (on the VLTS graphs) to more complex response properties that must be encoded into boolean equation systems with several blocks. Compared to the sequential version of EVALUATOR 3.5, the distributed version shows quasi-linear speedups, a good load balancing, and a low memory overhead. As regards deadlock and livelock detection, it compared favourably with UPPDMC [49], another distributed model checker for modal μ-calculus developed at RWTH (Aachen, Germany). A paper on this work was accepted for publication [27].
- We implemented a prototype connection of REDUCTOR's τ-confluence reduction algorithm to the distributed version of CÆSAR_SOLVE. Experiments were performed on the IDPOT cluster using various graphs taken from CADP demo examples. Each experiment consisted in generating a reduced graph using both the sequential version (based on algorithm A2 of CÆSAR_SOLVE) and the distributed version. We observed that the latter was faster by up to three orders of magnitude, with a low memory overhead. For some examples, the distributed version succeeded where the sequential one would fail due to memory exhaustion [17].
- Along the lines of the test generation theory [51] implemented in the TGV tool of CADP, we developed a prototype tool named EXTRACTOR that takes as inputs both a "specification" graph (represented implicitly using the OPEN/CÆSAR environment) and a "test purpose" (represented explicitly as a BCG graph), and computes the "complete test graph" (CTG) containing all sequences of observable actions and quiescence present in the specification and allowed by the test purpose. The CTG produced by EXTRACTOR is subsequently processed using the DETERMINATOR tool of CADP to eliminate nondeterminism and τ-transitions. Compared to TGV, EXTRACTOR uses a radically different approach, as it reformulates the CTG generation problem in terms of a boolean equation system, for which a diagnostic is computed using the CÆSAR_SOLVE library.
  We developed two versions of EXTRACTOR, a sequential one ($1,200$ lines of C code) based on the sequential resolution algorithms of CÆSAR_SOLVE, and a distributed one ($1,300$ lines of C code) based on the distributed version of CÆSAR_SOLVE. Experiments were performed on various graphs (taken from the VLTS benchmark suite and the CADP demo examples) by using generic test purposes expressing the reachability of certain visible actions. All CTGs obtained by applying EXTRACTOR and DETERMINATOR were strongly equivalent to those produced by TGV, although slightly larger. On some examples, however, the generation of the CTG succeeded using EXTRACTOR and DETERMINATOR, whereas TGV would fail because of memory shortage. These results have been accepted for publication [27].

5.1.7. Other Tool Developments

Participants: David Champelovier, Damien Bergamini, Hubert Garavel, Frédéric Lang, Radu Mateescu, Wendelin Serwe.

Late 2004 and early 2005, a significant number of new tools and libraries were integrated to the CADP toolbox, among which BCG_MERGE, BCG_STEADY, BCG_TRANSIENT, BISIMULATOR, CAESAR_AREA, CAESAR_MASK, CÆSAR_SOLVE, DETERMINATOR, DISTRIBUTOR, EVALUATOR 3.5, and PROJECTOR 2.0. This also implied an important effort in writing the corresponding manual pages and correcting the bugs reported by users worldwide.

Additionally, we improved the following CADP tools and libraries:

- The CAESAR_HASH library was improved by adding new hash functions and rewriting several existing ones.

- The CAESAR_TABLE library was enhanced by extending the table maximal capacity from $(2^{24}) - 1$ to 2^{29} elements, which increases the memory cost of a table, but in a reasonable manner. We also reduced (up to a factor of 2) the memory cost for "small" tables, the size of which can be known statically.
- The ALDÉBARAN tool, no longer maintained by its authors, was replaced by a shell wrapper (680 lines of shell-script) that invokes the new CADP tools BCG_MIN, BISIMULATOR, REDUCTOR 4.0, and EXP.OPEN 2.0 transparently, while keeping exactly the same command-line interface as the old ALDÉBARAN tool. Except in a few cases (graph comparison and minimization modulo delay equivalence, and minimization modulo observational equivalence), the old ALDÉBARAN tool is no longer used, which is a way to avoid 24 known bugs in this tool.
- We updated most of the CADP demo examples in order to take advantage of recent features and tools of CADP.

We have continued adapting CADP to the latest computing platforms:

- We ported CADP to the most recent LINUX distributions FEDORA CORE 3 and 4.
- We upgraded the WINDOWS version of CADP to support recent versions of MICROSOFT's MSVCRT and MINGWIN's W32API libraries.
- We finished porting the CADP tools with a graphical user-interface to the MAC OS X operating system and we took provisions to support its most recent version 10.4 "TIGER".
- Besides CADP, F. Lang updated the source code of the RTL timed verification tool developped by Christophe Lohr (formerly at LAAS/CNRS) to make it accepted by recent C++ compilers.

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 [2] for the data type part of LOTOS, the CÆSAR compiler [10] for the process part of LOTOS, and the CÆSAR.INDENT pretty-printer.

In 2005, we performed maintenance activities for these tools (1 bug fixed in CÆSAR.ADT, 1 bug fixed in CÆSAR, and 3 bugs fixed in CÆSAR.INDENT) and we improved the C code generated by CÆSAR and CÆSAR.ADT to avoid warnings emitted by the most recent C compilers. We also enhanced the CÆSAR compiler in two ways:

- In the framework of the FORMALFAME PLUS contract (see § 6.2), we simplified the use of the EXEC/CÆSAR environment [13]. EXEC/CÆSAR allows to interconnect, on the one hand, the C code generated by CÆSAR from the LOTOS description of a system and, on the other hand, the "real" environment with which this system interacts. This interconnection is implemented as a collection of C functions, one per visible gate declared in the LOTOS specification, which have to be written by hand.
  The new version of CÆSAR greatly automates this task by generating automatically, for each function, a C code skeleton that implements appropriate pattern-matching actions for checking gate parameters — since, in LOTOS, the same gate can be overloaded with several parameter lists that differ in number, types and direction (input or output) — as well as logging actions to trace the execution of these functions.

- We pursued our study of state space reduction techniques, our goal being to decrease the size of the graphs generated by CÆSAR, still preserving strong bisimulation between the original and reduced graphs.
  Our previous work on state space reduction based on live variable analysis [9] led to an improved version of CÆSAR (named CÆSAR.NEW), which became part of CADP in April 2005. A journal paper was also accepted for publication [19].
  Additionally, W. Serwe experimented further uses of data-flow analysis so as to reduce memory requirements for enumerative verification.

5.2.2. Compilation of E-LOTOS

Participants: David Champelovier, Hubert Garavel, Frédéric Lang.

As regards the E-LOTOS language — and, more specifically, its LOTOS NT variant elaborated by the VASY team — we worked in two directions:

- We continued to improve the 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 team as a development tool for compiler construction [7].
  In 2005, we released a new version 2.5 of TRAIAN. It corrects four bugs and makes the C code generated by TRAIAN compatible with the latest versions of GCC and Intel's ICC compilers. In addition, the TRAIAN libraries and shell-scripts have been ported to the ITANIUM 64-bit platform running the LINUX operating system.
- In the framework of the FORMALFAME PLUS contract (see § 6.2), we undertook the development of a translator from LOTOS NT to LOTOS, so as to ease the development of large specifications by BULL and to reuse the existing LOTOS tools for analyzing concurrent systems described in LOTOS NT.
  In 2005, a first version of this translator was delivered to BULL. It consists of a LOTOS preprocessing tool named LPP ($1,280$ lines of C code) and a translation tool named LNT2LOTOS developed using the aforementioned SYNTAX/TRAIAN technology (760 lines of SYNTAX code, $1,920$ lines of LOTOS NT code, and 980 lines of C code). A reference manual was written [33].

5.2.3. Source-Level Translations between Process Algebras

Participants: Hubert Garavel, Gwen Salaün, Wendelin Serwe.

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. 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. One first example is the aforementioned translator from LOTOS NT to LOTOS (see § 5.2.2). In 2005, we have also been studying translators for two other process algebras:

- We considered the process algebra FSP (*Finite State Processes*) defined in a popular textbook on concurrency [53]. For the "basic FSP" fragment (i.e., FSP without its data part), a prototype translator to LOTOS (700 lines of SYNTAX code, $2,300$ lines of LOTOS NT code, and 300 lines of C code) was developed. While extending this translator to "full FSP", ambiguities were found in the reference FSP grammar. A collaboration with Jeff Kramer and Jeff Magee (Imperial College, London) was initiated to handle these issues.

- In the framework of the INRIA/LETI collaboration (see § 7.1), we focused on the process algebra CHP (*Communicating Hardware Processes*) for which the TIMA laboratory has developed a circuit synthesis tool named TAST [58] and which is used by the LETI laboratory to describe complex, asynchronous circuits at a high abstraction level. The goal is to integrate formal verification into the design flow of complex microelectronic circuits.
  First, we defined a structural operational semantics for CHP, which so far lacked a formal semantics. In particular, our semantics gives an unambiguous meaning to the hardware-specific "probe" operator of CHP, the semantics of which has been debated for long beforehand.
  We then proposed a translation scheme from CHP to LOTOS for a fragment of CHP restricted to simple data types (booleans and natural numbers) and to one single probe operator in boolean guards. For this fragment we developed a prototype translator named CHP2LOTOS, which we used successfully to verify an asynchronous circuit implementing the DES encryption standard (see § 5.3). The operational semantics and the translation scheme for this CHP fragment led to an international publication [30].
  We then revised our translation scheme to handle all the data types of CHP (including vectors, arrays, and enumerated types of arbitrary size) and to allow boolean guards containing several probe operators. The CHP2LOTOS translator was extended accordingly (currently, $2,100$ lines of SYNTAX code, $11,500$ lines of LOTOS NT code, and $4,000$ lines of C code) and started to be applied to an asynchronous NOC (*Network on Chip*) circuit under design at the LETI laboratory (see § 5.3).

5.3. Case Studies and Practical Applications

Participants: David Champelovier, Hubert Garavel, Frédéric Lang, Radu Mateescu, Gwen Salaün, Wendelin Serwe.

In 2005, the VASY team also worked on the following case studies:

- We continued our collaboration with Antonella Chirichiello (University "La Sapienza", Rome) on the use of process algebras as a convenient design formalism for WEB services. This led to a new publication [24] on the use of CADP for the verification of an e-business application specified in the standard orchestration language BPEL and translated to LOTOS.
- In the context of the INRIA/LETI collaboration (see § 7.1), we pursued the study (undertaken in 2004) of an asynchronous circuit, designed by the LETI and TIMA laboratories, which implements the DES (*Data Encryption Standard*). We applied our CHP2LOTOS translator (see § 5.2.3) to a description of this circuit given in the CHP process algebra ($1,700$ lines) and the translator produced a LOTOS description of $3,800$ lines.
  Because of the high degree of concurrency in this circuit (25 concurrent processes), direct generation of the state space was not appropriate (more than 17 million states and 139 million transitions). However, the compositional verification techniques of CADP (see § 5.1.5) allowed to generate a smaller, yet equivalent state space ($16,910$ states and $85,840$ transitions) in 8 minutes, on which we verified several properties (absence of deadlocks, correct number of iterations, correct synchronisation between iterations).
- Also in the context of the INRIA/LETI collaboration, we started working on another circuit developed by the LETI laboratory, namely the asynchronous communication interconnect of a NOC (*Network on Chip*) described in CHP [41]. Our first results are encouraging: using our CHP2LOTOS translator, we were able to find several small mistakes in the CHP description.

- We continued the work undertaken in collaboration with Grégory Batt, Hidde de Jong, and Delphine Ropers (HELIX team of INRIA Rhône-Alpes) for connecting the GNA (*Genetic Network Analyzer*) tool developed by HELIX with CADP in order to verify temporal properties of genetic regulatory networks.
  GNA provides a simulator of qualitative models of genetic regulatory networks in the form of piecewise-linear differential equations. The output of the simulator is a Kripke structure, i.e., a state-transition graph in which the relevant information is associated to states. We defined a translation from Kripke structures to labeled transition systems (the graphs used by CADP) that preserves strong bisimulation and is succinct, i.e., the produced labeled transition system has the same number of states and transitions as the Kripke structure. This translation was implemented as a back-end of the GNA simulator, which became in this way directly connected to CADP.
  We also defined a translation from propositional μ-calculus to modal μ-calculus (the temporal logics used to express properties on Kripke structures and labeled transition systems, respectively) that preserves the truth of formulas. In conjunction with the translation between Kripke structures and labeled transition systems, this enabled to use the model checkers XTL and EVALUATOR 3.5 of CADP for verifying various temporal properties of genetic regulatory networks. It is worth noticing that certain properties (e.g., the presence of oscillations of protein concentrations), expressible in the μ-calculus fragment of alternation depth 2 but not in CTL, could not be verified using the NUSMV model checker, but were handled successfully using XTL. These activities led to two publications [22], [18].

A number of case-studies tackled by VASY during the past years have been finalized and properly integrated in CADP, which makes them available widely:

- a randomized binary distributed consensus protocol,
- a computer integrated manufacturing architecture,
- a distributed summation algorithm,
- a distributed Erathostene's sieve,
- a trader process for open distributed processing,
- a turntable system for drilling products, and
- an asynchronous circuit implementing the DES encryption standard.

Other teams also used the CADP toolbox for various case studies. To cite only recent work, we can mention:

- the verification of a reliable large scale multipoint transmission protocol combining terrestrial transmission with transmission via satellites [60],
- the analysis of an industrial manufacturing system [42],
- the behavioural verification of service composition [38],
- the modeling and verification of hierarchical components [39], [40],
- the generation of conformance tests for radiotherapy accelerators [59], and
- the use of LOTOS for constraint solving [55].

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 CHP2IF tool, developed by Menouer Boubekeur (TIMA laboratory, Grenoble), which allows the verification of asynchronous hardware via a translation of CHP descriptions to networks of communicating automata.
- the TTOOL tool, developed by Ludovic Apvrille (ENST, LABSOC laboratory, Sophia-Antipolis), which allows the verification of reachability graphs of UML diagrams using the TURTLE UML real-time profile.

6. Contracts and Grants with Industry

6.1. The IST ArchWare European Contract

Participants: David Champelovier, Hubert Garavel, Frédéric Lang, Radu Mateescu, Wendelin Serwe.

ARCHWARE (*Architecting Evolvable Software*) is a project of the European "Information Society Technologies" program (IST-2001-32360). Started on January 1st, 2002, ARCHWARE gathers the Research Consortium of Pisa (CPR), the Engineering company (Italy), the University of Savoie (LISTIC laboratory and "Association Interaction Université-Economie" — INTERUNEC), the THÉSAME company (France), the Universities of Manchester and St Andrews (United Kingdom), and the VASY team of INRIA.

The aim of ARCHWARE is to build an integrated environment for architecting evolvable software systems with functional and performance requirements [57].

In this context, VASY contributed to the definition of AAL (*Architecture Analysis Language*), a language dedicated to the description of behavioral properties of software architectures. AAL contains operators borrowed from first-order logic and modal μ-calculus, extended with predicates specific to architectural descriptions. It allows to specify both style-related structural properties (e.g., connectivity between components, cardinality, etc.) and architecture-related behavioral properties (e.g., safety, liveness, fairness).

VASY identified a fragment of AAL expressive enough for a large number of property patterns relevant to software architectures and developed a model checker for this fragment. This model checker translates the temporal formulas into boolean equation systems, which are solved on the models produced by the execution of the ARCHWARE virtual machine; the model checker is also equiped with diagnostics generation facilities.

Initially planned to terminate at the end of 2004, ARCHWARE was extended until June 30, 2005, this additional period being mainly devoted to integration and maintenance activities, dissemination, and preparation of the project final review. From our participation to ARCHWARE, we draw two main conclusions:

- The compiler construction technology promoted by VASY [7] proved to be effective for the development of the AAL model checker ($16,400$ lines of code).
- The verification technology produced by VASY [54] was successfully applied to AAL and allowed to check complex correctness properties on large event traces produced by the ARCHWARE virtual machine.

6.2. The FormalFame Plus Contract

Participants: Damien Bergamini, David Champelovier, Hubert Garavel, Radu Mateescu, Wendelin Serwe.

There is a long-standing collaboration between VASY and BULL, which aims at demonstrating that the formal methods and tools developed at INRIA can be successfully applied to BULL's multiprocessor architectures. The objective is to develop a complete and integrated solution supporting formal specification, simulation, rapid prototyping, verification, and testing.

Between 1995 and 1998, two case studies were successfully tackled using CADP: the POWERSCALE bus arbitration protocol [43] and the POLYKID multiprocessor architecture [13]).

Between 1998 and 2004, the collaboration focused on FAME, the CC-NUMA multiprocessor architecture used in BULL's NOVASCALE series of high-performance servers based on INTEL ITANIUM processors. The CADP tools have been used to validate a crucial circuit of FAME – the FSS (*Fame Scalability Switch*) – that implements the cache coherency protocol. The technology transfer is complete, in the sense that the CADP tools are now part of BULL's validation methodology and that BULL maintains itself the LOTOS specifications developed for FAME.

In 2004, the collaboration was renewed by a followup contract named FORMALFAME PLUS, which, in 2005, was extended for two more years. The general goal of FORMALFAME PLUS is to enhance the performance and usability of the CADP tools in prevision of the next multiprocessor architectures under design at BULL.

In 2005, the contributions of VASY were the following:

- A new functionality was added to the CÆSAR compiler, which allows to generate code skeletons automatically for the C functions that, in the EXEC/CÆSAR software [13], connect the C code generated by CÆSAR from the LOTOS description of a system to "real" environment with which the system interacts (see § 5.2.1). This will ease the task of writing such interface functions.
- We undertook the definition of an automatic translator from LOTOS NT to LOTOS (see § 5.2.3). This will allow BULL to develop formal models in a faster way, as LOTOS NT is more concise than LOTOS and closer to mainstream programming languages.

6.3. The Topcased project

Participants: Hubert Garavel, Frédéric Lang, Nathalie Lépy.

TOPCASED (*Toolkit in OPen-source for Critical Application and SystEms Development*) is a project of AESE, the French *pôle de compétitivité* 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 environment providing methods and tools for embedded system development, ranging from system and architecture specifications to software and hardware implementation through equipment definition.

In 2005, the VASY team contributed to TOPCASED as regards the combination of model-driven engineering and formal methods for asynchronous systems. H. Garavel is the INRIA representative at the TOPCASED executive committee, as well as the secretary of this committee. H. Garavel and F. Lang gave several tutorials and demonstrations of the CADP tools to the TOPCASED participants. Finally, N. Lépy attended a 5-day training session on ECLIPSE organized at AIRBUS (Toulouse, France).

6.4. Forthcoming Projects

Participants: Hubert Garavel, Frédéric Lang, Radu Mateescu, Wendelin Serwe.

In 2005, the VASY team contributed to the preparation of two future projects:

- OPENEMBEDD is a French national project of 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. Officially approved in 2005, OPENEMBEDD will start in January 2006 for three years.
- MULTIVAL (*Validation of Multiprocessor Multithreaded Architectures*) is a project proposed in the framework of MINALOGIC, the French *pôle de compétitivité* dedicated to micro-nano technologies and embedded software for systems on chip. MULTIVAL addresses verification and performance evaluation issues for three innovative asynchronous architectures developed by BULL, CEA/LETI, and ST MICROELECTRONICS. In December 2005, MULTIVAL was officially approved by MINALOGIC as part of its EMSOC/*Atelier du Futur* program.

7. Other Grants and Activities

7.1. National Collaborations

The VASY team plays an active role in the joint research center launched in 2004 between INRIA Rhône-Alpes and the LETI laboratory of CEA-Grenoble. In co-operation with LETI scientists (Edith Beigné, François Bertrand, Fabien Clermidy, Yvain Thonnart, and Pascal Vivet), VASY develops 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*). The TIMA laboratory (Dominique Borrione and Marc Renaudin) also contributes to this research action. In 2005, our work focused on a translator that connects the verification tools developed by VASY to the hardware synthesis tools developed by TIMA (see § 5.2.3).

Together with the OASIS team of INRIA Sophia-Antipolis (Tomas Barros and Eric Madelaine), the LTCI team of ENST-Paris (Hamid Irfan, Elie Najm, and Sylvie Vignes), the SVF team of the LAAS/CNRS laboratory (Bernard Berthomieu and François Vernadat), and the MVR team of IRIT (Mamoun Filali), VASY is part of the national action FIACRE – ACI *Sécurité Informatique* started in 2004 (see http://www-sop.inria.fr/oasis/fiacre). In 2005, we investigated semantic interconnections between the CADP toolbox and the tools developed by the other FIACRE partners.

Additionally, we collaborated in 2005 with several INRIA teams:

- HELIX (Rhône-Alpes): applications of model checking to biological systems (Grégory Batt, Delphine Ropers, and Hidde de Jong);
- OASIS (Sophia-Antipolis): collaboration in the framework of FIACRE national action (Tomas Barros and Eric Madelaine);
- POP ART (Rhône-Alpes): combination of the CADP and PROMETHEUS compositional verification tools (Gregor Goessler);
- WAM (Rhône-Alpes): application of satisfiability of the modal μ-calculus to optimize XPATH search queries on XML documents (Pierre Genevès and Nabil Layaïda).

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

- ID-IMAG laboratory (Montbonnot): use of the IDPOT cluster to experiment parallel and distributed verification algorithms (see § 5.1.6);
- LAAS-CNRS laboratory (Toulouse): collaboration in the framework of FIACRE national action, TOPCASED project, and forthcoming OPENEMBEDD project (Bernard Berthomieu and François Vernadat);
- LAMI laboratory (Evry) and Ecole des Mines de Nantes: coordination, adaptation, and analysis of component systems (Pascal Poizat and Jean-Claude Royer);
- LETI laboratory of CEA-Grenoble: collaboration in the framework of the INRIA/LETI joint research center and of the forthcoming MULTIVAL project (Edith Beigné, François Bertrand, Fabien Clermidy, Yvain Thonnart, and Pascal Vivet);
- LIP laboratory (Lyon) and INRIA Rhône-Alpes: between April and October 2005, R. Mateescu was hosted by the ARENAIRE team and, since October 2005, he has a part-time (20%) collaboration with the PLUME team.

7.2. International Collaborations

The VASY team of INRIA and the SEN2 team of CWI collaborate in SENVA, a joint research team on safety-critical systems (see http://www.inrialpes.fr/vasy/senva). Launched in 2004, the SENVA team is supported by INRIA's European and International Affairs Department and by CWI.

The VASY 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. Within FMICS, R. Mateescu contributes to the preparation of a "Formal Methods Handbook".

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, launched in 2005 and chaired by Luca Aceto.

H. Garavel is a member of the technical committee (*ETItorial Board*) of the ETI (*Electronic Tool Integration*) software development platform (see http://eti.cs.uni-dortmund.de).

In addition to our partners in aforementioned contractual collaborations, we had scientific relations in 2005 with several international universities and research centers, including:

- Eindhoven University of Technology (Judi Romijn and Xing Huo),
- Imperial College (Jeff Kramer and Jeff Magee),
- University of Konstanz (Husain Aljazzar and Paiam Salavati),
- University of Málaga (Carlos Canal and Pedro Merino) [31], and
- University "La Sapienza" of Rome (Antonella Chirichiello and Benjamin Habegger) [24].

7.3. Visits and Invitations

In 2005, we had the following scientific exchanges:

- Jean-Luc Nougaret and Franck Di Maio (CERN, Geneva, Switzerland) visited us on January 20, 2005.
- Benjamin Habegger (INRIA Futurs, MOSTRARE team) visited us on May 10–13, 2005.
- Pascal Poizat (University of Evry – Val d'Essonne) visited us on May 23, 2005.
- The annual SENVA seminar was held in St. Pierre de Chartreuse on May 30–June 1st, 2005. In addition to the VASY team, Wan Fokking (Free University of Amsterdam), Jeff Kramer and Jeff Magee (Imperial College, London), Aad Mathijssen (University of Eindhoven), Jaco van de Pol and Anton Wijs (CWI, Amsterdam), Mihaela Sighireanu (University of Paris 7), and Michael Weber (RWTH Aachen) attended this seminar. The list of talks is available from http://www.inrialpes.fr/vasy/senva/workshop2005.
- In the framework of the FIACRE national action, we organized a meeting at INRIA Rhône-Alpes on September 26–27, 2005 attended by the following visitors: Tomas Barros and Eric Madelaine (INRIA Sophia Antipolis), Irfan Hamid, Elie Najm, and Sylvie Vignes (ENST Paris), Mamoun Filali and François Vernadat (FERIA/CNRS, Toulouse), and Jean-Bernard Stefani (INRIA Rhône-Alpes).
- In the framework of the SENVA collaboration, we organized an international meeting on "*Clusters and Grids for Verification and Performance Evaluation*" held at INRIA Rhône-Alpes on November 16–17, 2005. In addition to the VASY team, this meeting was attended by Jiri Barnat, Lubos Brim, and Ivana Cerna (Masaryk University Brno), Gerd Behrmann and Josva Kleist (Aalborg University), Anne Benoit (INRIA, GRAAL team, Lyon), Stefan Blom (Innsbruck University), François Brown de Colstoun (INRIA), Boudewijn Haverkort (University of Twente), William Knottenbelt and Tamas Suto (Imperial College, London), Marta Kwiatkowska (University of Birmingham), Matthias Kuntz (Universität der Bundeswehr, Munich), Martin Leucker (Technical University of Munich), Simona Orzan (Technical University of Eindhoven), Jaco van de Pol (CWI, Amsterdam), and Michael Weber (RWTH Aachen). The list of talks is available from http://www.inrialpes.fr/vasy/senva/meeting2005.

8. Dissemination

8.1. Software Dissemination and Internet Visibility

The VASY team distributes two main software tools: the CADP toolbox (see § 4.1) and the TRAIAN compiler (see § 4.2). In 2005, the main facts are the following:

- We prepared and distributed 15 successive beta-versions (2003-s, ..., 2003-z, 2004-a, ..., 2004-g) of CADP.
- The number of license contracts signed for CADP increased from 330 to 345.
- We were requested to grant CADP licenses for 663 different computers in the world.
- The distribution of the TRAIAN compiler continued and a new version 2.5 of TRAIAN (see § 5.2.2) was released on October 6, 2005.
- The TRAIAN compiler was downloaded by 51 different sites.

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

8.2. Program Committees

In 2005, the members of VASY assumed the following responsibilities:

- H. Garavel was, together with John Hatcliff (Kansas State University), responsible for a special issue of the TCS (*Theoretical Computer Science*) journal, to appear in 2006, which gathers the best theory-oriented papers of TACAS'2003.
- H. Garavel was, together with John Hatcliff (Kansas State University), responsible for a special issue of the STTT (*Software Tools for Technology Transfer*) journal, to appear in 2006, which gathers the best software-oriented papers of TACAS'2003.
- H. Garavel was a steering committee member of PDMC (*Parallel and Distributed Methods in Verification*) series of international workshops.
- H. Garavel was a program committee member of PDMC'2005 (*4th International Workshop on Parallel and Distributed Methods in VerifiCation*, Lisbon, Portugal, July 10, 2005).
- H. Garavel was a program committee member of SOFTMC'2005 (*3rd International Workshop on Software Model Checking*, Edinburgh, Scotland, United Kingdom, July 11, 2005).
- R. Mateescu was a program committee member of TACAS'2005 (*11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Edinburgh, Scotland, United Kingdom, April 4-8, 2005).
- R. Mateescu was a program committee member of VVEIS'2005 (*3rd International Workshop on Verification and Validation of Enterprise Information Systems*, Miami, Florida, USA, May 13, 2005).
- R. Mateescu was a program committee member of EWSA'2005 (*2nd European Workshop on Software Architecture*, Pisa, Italy, June 13-14, 2005).
- R. Mateescu was a program committee member of FMICS'2005 (*10th International Workshop on Formal Methods for Industrial Critical Systems*, Lisbon, Portugal, September 5-6, 2005).
- R. Mateescu was a program committee member of ETR'2005 (*Ecole d'été temps réel 2005*, Nancy, France, September 13-16, 2005).

8.3. Lectures and Invited Conferences

In 2005, we gave talks in several international conferences and workshops (see bibliography below). Additionally:

- R. Mateescu gave a talk entitled "*Vérification à la volée de systèmes parallèles asynchrones*" at the LIP laboratory – INRIA Rhône-Alpes (Lyon, France) on February 1st, 2005.
- G. Salaün gave a talk entitled "*Describing and Reasoning on Web Services using Process Algebra*" at INRIA Lorraine (Nancy, France) on February 7, 2005.
- W. Serwe participated to the "*First German Verification Day*" (Oldenburg, Germany) on March 4, 2005.
- G. Salaün gave a talk entitled "*Describing and Reasoning on Web Services using Process Algebra*" at INRIA Rennes (France) on March 17, 2005.
- G. Salaün gave a talk entitled "*Describing and Reasoning on Web Services using Process Algebra*" at Ecole des Mines de Nantes (France) on April 4, 2005.
- F. Lang gave a talk entitled "*Verification of the ODP Trader using* EXP.OPEN *2.0 and* CADP" at the LAAS/CNRS laboratory (Toulouse, France) on April 18–19, 2005.
- G. Salaün gave two talks entitled "*Describing and Reasoning on Web Services using Process Algebra*" and "*Formal Coordination of Distributed Entities Described with Behavioural Interfaces*" at the LAAS/CNRS laboratory (Toulouse, France) on April 18–19, 2005.
- C. Joubert gave a talk entitled "*Distributed On-the-Fly Verification of Finite-State Systems*" at the Technical University of Valencia (Spain) on May 9, 2005.
- R. Mateescu gave a talk entitled "*Résolution à la volée des systèmes d'équations booléennes et applications*" at the LSV laboratory (Cachan, France) on May 24, 2005.
- R. Mateescu gave a talk entitled "CÆSAR_SOLVE: *A Generic Library for On-the-Fly Resolution of Boolean Equation Systems and its Applications to Verification*" at the University of Málaga (Spain) on June 27, 2005.
- H. Garavel gave a tool demonstration entitled "*DISTRIBUTOR and BCG_MERGE: Tools for Distributed Explicit State Space Generation*" at PDMC'2005 (*4th International Workshop on Parallel and Distributed Methods in VerifiCation*, Lisbon, Portugal) on July 10, 2005.
- H. Garavel gave an invited talk entitled "*How to Interface Algebraic Process Calculi with the Real World?*" at the international seminar "*Algebraic Process Calculi: The First Twenty Five Years and Beyond*" held in Bertinoro (Forlí, Italy) on August 1–5, 2005.
- R. Mateescu gave a public demonstration of CADP at the summer school "*Ecole d'été temps réel*" (Nancy, France) on September 14, 2005.
- F. Lang gave a public demonstration of CADP at a meeting of the TOPCASED project (LAAS/CNRS, Toulouse, France) on September 1, 2005.
- F. Lang and W. Serwe visited CWI (Amsterdam, The Netherlands) on September 12–16, 2005:
  - F. Lang gave a talk entitled "EXP.OPEN *2.0: A Flexible Tool Integrating Partial Order, Compositional and On-the-fly Verification Methods*" at the PAM (*Process Algebra Meeting*) held at CWI on September 14, 2005.
  - W. Serwe gave a talk entitled "*State Space Reduction for Process Algebra Specifications*" at the PAM (*Process Algebra Meeting*) held at CWI on September 14, 2005.

- G. Salaün gave a talk entitled "*How Formal Methods Can Contribute to the Formal Development of Web Services*" at the LAMI laboratory (Evry, France) on September 19, 2005.
- F. Lang participated to the TOPCASED Industrial Workshop on System Verification held at AIRBUS (Toulouse, France) on October 11, 2005, where he gave a talk entitled "*Description des comportements synchrones et asynchrones*" and demonstrated the CADP toolbox.
- W. Serwe represented INRIA during a visit, organized by the French Embassy in Tokyo, of Japanese public and industrial research institutes working in the field of systems on chip (Tokyo, Japon, November 7–11, 2005).
- H. Garavel gave an invited talk entitled "*An Overview of* CADP *2005*" at the German Transregional Collaborative Research Center AVACS (*Automatic Verification and Analysis of Complex Systems*) in Freiburg (Germany) — simultaneously transmitted to Oldenburg and Saarbrücken — on November 25, 2005.
- H. Garavel gave a talk entitled "*Systèmes asynchrones, algèbres de processus et espaces d'états*" at the LIP laboratory – INRIA Rhône-Alpes (Lyon, France) on December 13, 2005.
- C. Joubert gave a talk entitled "*Distributed On-the-Fly Verification of Large State Spaces*" at the University of Málaga (Spain) on December 22, 2005.

8.4. Teaching Activities

The VASY team is a host team for:

- The computer science master entitled "*Informatique : Systèmes et Logiciels*", common to Institut National Polytechnique de Grenoble and Université Joseph Fourier,
- The computer science master entitled "*Informatique : communication et coopération dans les systèmes à agents*" of Université de Savoie.

In 2005:

- F. Lang and W. Serwe gave the course on "*Temps Réel*" to the 3rd year students of ENSIMAG (18 hours).
- C. Joubert gave a course on "*Tools for Software Engineering*" to the 4th year students of Université Joseph Fourier (9 hours).
- C. Joubert gave lectures and programming assignments for the "*Formal Specification*", "*Computer Networks*", "*Software Architecture*", and "*Operating Systems*" courses at Université Joseph Fourier (87 hours).
- H. Garavel supervised the internship (*mémoire de probatoire* CNAM) of Vincent Doucet entitled "*Vérification distribuée de programmes parallèles*", defended in Grenoble on March 31, 2005.
- R. Mateescu was a jury member of Loïc Strus' MSc thesis (DEA) entitled "*Test de propriétés*", defended at the University Joseph Fourier (Grenoble) on June 22, 2005.
- R. Mateescu was a jury member of Jesús Martínez Cruz's PhD thesis entitled "*Un enfoque basado en estándares para la integración de técnicas y herramientas de Ingeniería de Protocolos*", defended at the University of Málaga (Spain) on June 28, 2005.
- F. Lang supervised the internship (*mémoire de probatoire* CNAM) of N. Lépy entitled "*Etude de l'environnement ouvert de développement intégré* ECLIPSE *dans l'optique d'une extension*", defended in Grenoble on July 1st, 2005.
- F. Lang was a jury member of Arnaud Lanoix's PhD thesis entitled "*Systèmes à composants synchronisés : contributions à la vérification compositionnelle du raffinement et des propriétés*", defended at Université de Franche Comté (Besançon) on August 31, 2005.

- F. Lang was a jury member of Tomas Barros's PhD thesis entitled "*Formal specification and verification of distributed components*", defended at Université de Nice Sophia-Antipolis on November 25, 2005.
- H. Garavel and R. Mateescu supervised the PhD thesis of C. Joubert entitled "*Vérification distribuée à la volée de grands espaces d'états*", defended on December 12, 2005 [17].

8.5. Miscellaneous Activities

D. Champelovier participates to the design group for the new INRIA Rhône-Alpes WEB site.

H. Garavel is a member of the budget and computing facilities committees of INRIA Rhône-Alpes.

Within the EMSOC/*Atelier du Futur* program of the MINALOGIC *pôle de compétitivité*, H. Garavel is a member of the working group (6 persons) in charge of making proposals for governance and project selection.

F. Lang participates to the consultative organizational committee of INRIA Rhône-Alpes.

W. Serwe is a member of the continuous training committee of INRIA Rhône-Alpes.

9. Bibliography

Major publications by the team in recent years

[1] 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.

[2] 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, December 1989, p. 147–162.

[3] 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), Berlin", B. STEFFEN (editor). , Lecture Notes in Computer Science, Full version available as Inria Research Report RR-3352, vol. 1384, Springer Verlag, March 1998, p. 68–84, http://www.inria.fr/rrrt/rr-3352.html.

[4] 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, Full version available as Inria Research Report 4492, vol. 2391, Springer Verlag, July 2002, p. 410–429, http://www.inria.fr/rrrt/rr-4492.html.

[5] 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). , Full version available as Inria Research Report RR-4223, Kluwer Academic Publishers, IFIP, August 2001, p. 377–392, http://www.inria.fr/rrrt/rr-4223.html.

[6] 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, Full version available as Inria Research Report RR-4666, vol. 2529, Springer Verlag, November 2002, p. 276–291, http://www.inria.fr/rrrt/rr-4666.html.

[7] 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, vol. 2304, Springer Verlag, April 2002, p. 9–13.

[8] H. GARAVEL, R. MATEESCU, I. SMARANDACHE-STURM. *Parallel State Space Construction for Model-Checking*, in "Proceedings of the 8th International SPIN Workshop on Model Checking of Software SPIN'2001 (Toronto, Canada), Berlin", M. B. DWYER (editor). , Lecture Notes in Computer Science, Full version available as Inria Research Report RR-4341, vol. 2057, Springer Verlag, May 2001, p. 217–234, http://www.inria.fr/rrrt/rr-4341.html.

[9] H. GARAVEL, W. SERWE. *State Space Reduction for Process Algebra Specifications*, in "Proceedings of the 10th International Conference on Algebraic Methodology and Software Technology AMAST'2004 (Stirling, Scotland, UK)", C. RATTRAY, S. MAHARAJ, C. SHANKLAND (editors). , Lecture Notes in Computer Science, vol. 3116, Springer Verlag, July 2004, p. 164–180.

[10] 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). , North-Holland, IFIP, June 1990, p. 379–394.

[11] 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), Amsterdam", J.-F. GROOTE, B. LUTTIK, J. VAN WAMEL (editors). , Invited talk, CWI, May 1998, p. 187–230.

[12] 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). , Kluwer Academic Publishers, IFIP, October 1999, p. 185–202.

[13] H. GARAVEL, C. VIHO, M. ZENDRI. *System Design of a CC-NUMA Multiprocessor Architecture using Formal Specification, Model-Checking, Co-Simulation, and Test Generation*, in "Springer International Journal on Software Tools for Technology Transfer (STTT)", Full version available as Inria Research Report RR-4041, vol. 3, n° 3, July 2001, p. 314–331, http://www.inria.fr/rrrt/rr-4041.html.

[14] R. MATEESCU. *A Generic On-the-Fly Solver for Alternation-Free Boolean Equation Systems*, in "Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2003 (Warsaw, Poland)", H. GARAVEL, J. HATCLIFF (editors). , Lecture Notes in Computer Science, Full version available as Inria Research Report RR-4711, vol. 2619, Springer Verlag, April 2003, p. 81–96, http://www.inria.fr/rrrt/rr-4711.html.

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

[16] G. PACE, F. LANG, R. MATEESCU. *Calculating τ-Confluence Compositionally*, in "Proceedings of the 15th International Conference on Computer Aided Verification CAV'2003 (Boulder, Colorado, USA)", W. A. HUNT JR, F. SOMENZI (editors). , Lecture Notes in Computer Science, Full version available as INRIA Research Report RR-4918, vol. 2725, Springer Verlag, July 2003, p. 446–459, http://www.inria.fr/rrrt/rr-4918.html.

Doctoral dissertations and Habilitation theses

[17] C. JOUBERT. *Vérification distribuée à la volée de grands espaces d'états*, Thèse de Doctorat, Institut National Polytechnique de Grenoble, December 2005.

Articles in refereed journals and book chapters

[18] G. BATT, D. ROPERS, H. DE JONG, J. GEISELMANN, R. MATEESCU, M. PAGE, D. SCHNEIDER. *Validation of Qualitative Models of Genetic Regulatory Networks by Model Checking: Analysis of the Nutritional Stress Response in Escherichia Coli*, in "Bioinformatics", vol. 21, n° Suppl 1, 2005, p. i19–i28.

[19] H. GARAVEL, W. SERWE. *State Space Reduction for Process Algebra Specifications*, in "Theoretical Computer Science", to appear, 2006.

[20] F. LANG. *Explaining the Lazy Krivine Machine Using Explicit Substitution and Addresses*, in "Journal of Higher-Order and Symbolic Computation, special issue on Krivine's machine", to appear, 2006.

[21] R. MATEESCU. *CAESAR_SOLVE: A Generic Library for On-the-Fly Resolution of Alternation-Free Boolean Equation Systems*, in "Springer International Journal on Software Tools for Technology Transfer (STTT)", to appear, 2006.

Publications in Conferences and Workshops

[22] G. BATT, D. ROPERS, H. DE JONG, J. GEISELMANN, R. MATEESCU, M. PAGE, D. SCHNEIDER. *Analysis and Verification of Qualitative Models of Genetic Regulatory Networks: A Model-Checking Approach*, in "Proceedings of the 19th International Joint Conference on Artificial Intelligence IJCAI'05 (Edinburgh, Scotland)", L. P. KAELBLING, A. SAFFIOTTI (editors). , July–August 2005, p. 370–375.

[23] D. BERGAMINI, N. DESCOUBES, C. JOUBERT, R. MATEESCU. *BISIMULATOR: A Modular Tool for On-the-Fly Equivalence Checking*, in "Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2005 (Edinburgh, Scotland, UK)", N. HALBWACHS, L. ZUCK (editors). , Lecture Notes in Computer Science, vol. 3440, Springer Verlag, April 2005, p. 581–585.

[24] A. CHIRICHIELLO, G. SALAÜN. *Encoding Abstract Descriptions into Executable Web Services: Towards a Formal Development*, in "Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence WI'05 (Compiègne, France)", Extended version available as Technical Report 08-05 of Università di Roma "La Sapienza" (DIS department), IEEE Press, September 2005, p. 457–463.

[25] H. GARAVEL, R. MATEESCU, D. BERGAMINI, A. CURIC, N. DESCOUBES, C. JOUBERT, I. SMARANDACHE-STURM, G. STRAGIER. *DISTRIBUTOR and BCG_MERGE: Tools for Distributed Explicit State Space Generation*, in "Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2006 (Vienna, Austria)", H. HERMANNS, J.

PALBERG (editors). , Lecture Notes in Computer Science, to appear, Springer Verlag, March–April 2006.

[26] C. JOUBERT, R. MATEESCU. *Distributed Local Resolution of Boolean Equation Systems*, in "Proceedings of the 13th Euromicro Conference on Parallel, Distributed and Network-Based Processing PDP'2005 (Lugano, Switzerland)", F. TIRADO, M. PRIETO (editors). , IEEE Computer Society, February 2005, p. 264–271.

[27] 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, to appear, Springer Verlag, March–April 2006.

[28] F. LANG. *EXP.OPEN 2.0: A Flexible Tool Integrating Partial Order, Compositional, and On-the-fly Verification Methods*, in "Proceedings of the 5th International Conference on Integrated Formal Methods IFM'2005 (Eindhoven, The Netherlands)", J. VAN DE POL, J. ROMIJN, G. SMITH (editors). , Lecture Notes in Computer Science, Full version available as INRIA Research Report RR-5673, Springer Verlag, November 2005, http://www.inria.fr/rrrt/rr-5673.html.

[29] 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). , ACM Computer Society Press, ERCIM, September 2005, p. 80–89.

[30] G. SALAÜN, W. SERWE. *Translating Hardware Process Algebras into Standard Process Algebras — Illustration with CHP and LOTOS*, in "Proceedings of the 5th International Conference on Integrated Formal Methods IFM'2005 (Eindhoven, The Netherlands)", J. VAN DE POL, J. ROMIJN, G. SMITH (editors). , Lecture Notes in Computer Science, Full version available as INRIA Research Report RR-5666, Springer Verlag, November 2005, http://www.inria.fr/rrrt/rr-5666.html.

Internal Reports

[31] C. CANAL, P. POIZAT, G. SALAÜN. *Adaptation of Component Protocols using Synchronous Vectors*, ITI-05-10, University of Málaga, December 2005.

[32] P. POIZAT, G. SALAÜN. *Formal Coordination of Communicating Entities described with Behavioural Interfaces*, Research Report, nº 120-2005, LaMI, Evry, September 2005.

Miscellaneous

[33] D. CHAMPELOVIER, H. GARAVEL. *Reference Manual of the LOTOS NT to LOTOS Translator – Version 1D*, INRIA/VASY, 29 pages, November 2005.

[34] N. LÉPY. *Etude de l'environnement de developpement intégré ouvert Eclipse dans l'optique d'une extension*, Mémoire de probatoire en informatique, Conservatoire National des Arts et Métiers, Grenoble, July 2005.

[35] VASY. *Distributor Manual Page*, January 2005, http://www.inrialpes.fr/vasy/cadp/man/distributor.html.

[36] VASY. *Evaluator Version 3.5 Manual Page*, February 2005, http://www.inrialpes.fr/vasy/cadp/man/evaluator.html.

[37] VASY. *Reductor Version 4 Manual Page*, November 2005, http://www.inrialpes.fr/vasy/cadp/man/reductor.html.

Bibliography in notes

[38] P. ANDRÉ, G. ARDOUREL, C. ATTIOGBÉ. *Behavioural Verification of Service Composition*, in "Proceedings of the First International Workshop on Engineering Service Compositions WESC'2005 (Amsterdam, The Netherlands)", C. ZIRPINS, G. ORTIZ, W. LAMERDORF, W. EMMERICH (editors). , IBM Research Report RC23821, December 2005, p. 77–84.

[39] T. BARROS, L. HENRIO, E. MADELAINE. *Behavioural Models for Hierarchical Components*, in "Model Checking Software, Proceedings of the 12th International SPIN Workshop on Model Checking of Software SPIN'2005 (San Francisco, USA)", P. GODEFROID (editor). , Lecture Notes in Computer Science, vol. 3639, Springer Verlag, August 2005, p. 154–168.

[40] T. BARROS, L. HENRIO, E. MADELAINE. *Verification of Distributed Hierarchical Components*, in "Proceedings of the International Workshop on Formal Aspects of Component Software FACS'05 (Macao,)", Electronic Notes in Theoretical Computer Science, October 2005.

[41] 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, March 2005, p. 54–63.

[42] E. BORTNIK, N. TRCKA, A. J. WIJS, S. P. LUTTIK, J. M. VAN DE MORTEL-FRONCZAK, J. C. M. BAETEN, W. J. FOKKINK, J. E. RODA. *Analyzing a χ Model of a Turntable System using Spin, CADP and UPPAAL*, in "Journal of Logic and Algebraic Programming", vol. 65, n° 2, November–December 2005, p. 51–104.

[43] G. CHEHAIBAR, H. GARAVEL, L. MOUNIER, N. TAWBI, F. ZULIAN. *Specification and Verification of the PowerScale Bus Arbitration Protocol: An Industrial Experiment with LOTOS*, 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'96 (Kaiserslautern, Germany)", R. GOTZHEIN, J. BREDEREKE (editors). , Full version available as Inria Research Report RR-2958, Chapman & Hall, IFIP, October 1996, p. 435–450, http://www.inria.fr/rrrt/rr-2958.html.

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

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

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

[47] J. GROOTE, J. VAN DE POL. *State space reduction using partial τ-confluence*, in "Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science MFCS'2000 (Bratislava, Slovakia), Berlin", M. NIELSEN, B. ROVAN (editors). , Lecture Notes in Computer Science, Available as Cwi Technical Report SEN-R0008, Amsterdam, March 2000, vol. 1893, Springer Verlag, August 2000, p. 383–393.

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

[49] F. HOLMÉN, M. LEUCKER, M. LINDSTRÖM. *UppDMC – A Distributed Model Checker for Fragments of the μ-calculus*, in "Proceedings of the 3rd International Workshop on Parallel and Distributed Methods in Verification PDMC'2004 (London, UK)", L. BRIM, M. LEUCKER (editors). , Electronic Notes in Theoretical Computer Science, vol. 128, Elsevier Science Publishers, 2004, p. 91–105.

[50] ISO/IEC. *Enhancements to LOTOS (E-LOTOS)*, International Standard, n° 15437:2001, International Organization for Standardization — Information Technology, Genève, September 2001.

[51] C. JARD, T. JÉRON. *TGV: Theory, Principles and Algorithms*, in "Springer International Journal on Software Tools for Technology Transfer (STTT)", vol. 7, n° 4, 2005, p. 297–315.

[52] 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), Berlin", E. BRINKSMA (editor). , Lecture Notes in Computer Science, Extended version with proofs available as Research Report VERIMAG RR97-01, vol. 1217, Springer Verlag, April 1997.

[53] J. MAGEE, J. KRAMER. *Concurrency: State Models and Java Programs*, Wiley, 1999.

[54] R. MATEESCU. *Model Checking for Software Architectures*, in "Proceedings of the 1st European Workshop on Software Architecture EWSA'2004 (St Andrews, Scotland, UK)", F. OQUENDO, B. WARBOYS, R. MORRISON (editors). , Lecture Notes in Computer Science, vol. 3047, Springer Verlag, May 2004, p. 219–224.

[55] M. MOUHOUB, S. SADAOUI. *Improving LOTOS Simulation Using Constraint Propagation*, in "Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence ICTAI'05 (Hong-Kong)", November 2005.

[56] L. MOUNIER. *Méthodes de vérification de spécifications comportementales : étude et mise en œuvre*, Thèse de Doctorat, Université Joseph Fourier (Grenoble), January 1992, http://tel.ccsd.cnrs.fr/tel-00004729.

[57] F. OQUENDO, B. WARBOYS, R. MORRISON, R. DINDELEUX, F. GALLO, H. GARAVEL, C. OCCHIPINTI. *ArchWare: Architecting Evolvable Software*, in "Proceedings of the 1st European Workshop on Software Architecture EWSA'2004 (St Andrews, Scotland, UK)", F. OQUENDO, B. WARBOYS, R. MORRISON (editors). , Lecture Notes in Computer Science, Invited paper, vol. 3047, Springer Verlag, May 2004, p. 257-271.

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

[59] K. J. TURNER. *Test Generation for Radiotherapy Accelerators*, in "Springer International Journal on Software Tools for Technology Transfer (STTT)", vol. 7, n° 4, August 2005, p. 361–375.

[60] F. DE BELLEVILLE. *Transport multipoint fiable à très grande echelle : Intégration de critères de coût en environnement Internet hybride satellite/terrestre*, Ph. D. Thesis, Institut National Polytechnique de Toulouse, December 2004, http://tel.ccsd.cnrs.fr/tel-00008639.