INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team VASY

Validation of Systems

Rhône-Alpes

THEME 1C

Activity Report

2003

Table of contents

1. Team

Head of team
Hubert Garavel [DR2 INRIA]

Administrative Assistants
Valérie Gardès [until June 13, 2003]
Anne-Marie Saez [until February 4, 2003]
Catherine Magnin [since May 12, 2003]

Inria Staff
Radu Mateescu [CR1 INRIA]
Frédéric Lang [CR1 INRIA]

Bull Staff
Solofo Ramangalahy [BULL engineer]

Teaching Assistant
Aurore Collomb [Université Pierre Mendès France, until August 31, 2003]

Software Engineers
Damien Bergamini
David Champelovier
Nicolas Descoubes
Frédéric Tronel [until July 31, 2003]

Post-Doctoral Fellows
Aurore Collomb [since September 1st, 2003]
Wendelin Serwe [since March 10, 2003]

Ph. D. Student
Christophe Joubert

Student Interns
Alban Catry [Ecole Polytechnique (Palaiseau), from March 3 to September 30, 2003]
Guillaume Schaeffer [Supélec (Metz), from April 3 to September 12, 2003]

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 simple, 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 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 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 § 5.1) 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 § 5.1).
- Between 1992 and 2001, we contributed to the 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 from the WEB (see http://www.inrialpes.fr/vasy/elotos).
- We are also working on an E-LOTOS variant, named LOTOS NT (LOTOS *New Technology*) [9], 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 § 5.2).

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.

4. 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 § 6.3.4) 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.

5. Software

5.1. The CADP Toolbox

Participants: Damien Bergamini, David Champelovier, Nicolas Descoubes, Hubert Garavel [correspondent], 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 engineering of protocols and distributed systems (see http://www.inrialpes.fr/vasy/cadp). In this toolbox, we develop the following tools:

- CÆSAR.ADT [1] is a compiler that translates LOTOS abstract data types into C types and C functions. The translation involves pattern-matching compiling techniques and automatic recognition of usual types (integers, enumerations, tuples, etc.), which are implemented optimally.
- CÆSAR [8] is a compiler that translates LOTOS processes into either C code (for rapid prototyping and testing purposes) or finite graphs (for verification purpose). The translation is done using several intermediate steps, among which the construction of a Petri net extended with typed variables, data handling features, and atomic transitions.
- OPEN/CÆSAR [2] 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 provides a set of libraries with their programming interfaces, as well as various tools, such as:
  - EVALUATOR [17], 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 and REDUCTOR, which construct the graph of reachable states,
  - 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_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_MIN, which minimizes a graph modulo strong or branching equivalences (and can also deal with probabilistic and stochastic systems), and
  - BCG_OPEN, which allows to apply all OPEN/CÆSAR tools to any BCG graph.

- XTL (*eXecutable Temporal Language*) is a high level, functional language for programming algorithms that explore 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 allows to specify in XTL evaluation and diagnostic generation fixpoint algorithms for usual temporal logics (such as HML [44], CTL [41], ACTL [45], etc.)
- SVL (*Script Verification Language*) [4] is a scripting language to build complex verification scenarios that will, upon execution, invoke the appropriate CADP tools automatically.

The CADP toolbox also includes additional tools, such as those developed by the VERIMAG laboratory (Grenoble) and the VERTECS team of INRIA Rennes:

- ALDÉBARAN compares and minimizes graphs modulo various equivalence and preorder relations,
- EXP.OPEN 1.0 and PROJECTOR 1.0 compute products and abstractions of communicating automata, and
- TGV (*Test Generation based on Verification*) generates conformance tests according to user-defined test purposes.

All tools are integrated within the EUCALYPTUS graphical interface, which provides users with an easy, uniform access to the tools by hiding file formats and command-line syntax specific to each tool.

5.2. The TRAIAN Compiler

Participants: David Champelovier, Hubert Garavel [correspondent], 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 found useful applications in compiler construction [6]. The recent compilers developed by the VASY team — namely AAL (see § 6.1.5), EVALUATOR 4.0 (see § 6.1.4), EXP.OPEN 2.0 (see § 6.1.7), NTIF (see § 6.2.3), and SVL (see § 6.1.9) — 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 the 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).

6. New Results

6.1. Models and Verification Techniques

6.1.1. The BCG_GRAPH Tool

Participants: David Champelovier, Hubert Garavel, Frédéric Lang, Frédéric Tronel.

Designed to speed up the compositional verification of asynchronous systems (see § 6.3.2), BCG_GRAPH is a tool for generating the BCG graphs corresponding to FIFO communication buffers efficiently.

In 2003, we extended BCG_GRAPH so as to generate two other kinds of graphs:

- *Bag automata*, which model asynchronous communication buffers that do not preserve the ordering of messages. Each bag automaton is defined by the number and list of messages it may contain.

- *Chaos automata*, which are graphs containing one single state and and a set of looping transitions on that state. Each chaos automaton is defined by the list of its transition labels.

BCG_GRAPH (2, 700 lines of C code) allows to generate large BCG graphs (hundreds of thousands of states) within a few minutes. The generated graphs are always minimal modulo strong bisimulation.

6.1.2. 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 § 6.1.3), the model checker EVALUATOR 4.0 (see § 6.1.4), and the τ-confluence reduction tool (see § 6.1.6). The resolution method is based on boolean graphs, which provide an intuitive representation of dependencies between boolean variables; boolean graphs are handled implicitly in a way similar to the OPEN/CÆSAR interface.

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 2003, the CÆSAR_SOLVE library (7, 250 lines of C code) was extended and improved as follows:

- A new algorithm that generates tree diagnostics of minimal height was developed. This algorithm reduces by several orders of magnitude the height of the trees produced by depth-first search resolution algorithms (such as A1, A3, and A4), which is often prohibitive. Experiments indicate that the new algorithm also provides significant reductions for non-tree diagnostics (such as the counterexamples produced by equivalence checkers, which in general are directed acyclic graphs).
- The library was enhanced with a textual format for boolean equation systems. Primitives for reading/writing this format and an application tool that reads and solves boolean equation systems was developed.
- The library was enhanced with a primitive giving statistical information about the resolution (size of the boolean variables, number of boolean variables explored during resolution, etc.).
- The interface of the library was properly documented in view of its future integration within the OPEN/CÆSAR environment.

The CÆSAR_SOLVE library was subject to an invited publication [23].

6.1.3. The BISIMULATOR Tool

Participants: Nicolas Descoubes, 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. Thanks 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 accept lower level models (product of communicating automata).

BISIMULATOR works by reformulating the graph comparison problem in terms of a boolean equation system, which is solved using the CÆSAR_SOLVE library (see § 6.1.2). A useful functionality of BISIMULATOR is the generation of diagnostics (counterexamples), which explain why two graphs are not equivalent (or not included one in the other). The counterexamples generated by BISIMULATOR are directed acyclic graphs and usually much smaller than those generated by other tools (such as ALDÉBARAN) that can only generate counterexamples restricted to sets of traces.

In 2003, we continued the development of the BISIMULATOR tool ($10,000$ lines of C code):

- An optimized encoding of equivalence relations in terms of boolean equation systems was designed, which exploits local determinism to simplify the right-hand sides of boolean equations. This optimization increased the speed of BISIMULATOR by a factor between 2 and 5. It is worth noticing that boolean equations can be simplified on the fly: There is no need to check before verification if the explicit graph (represented in BCG format) is deterministic.
- The diagnostic generation of BISIMULATOR was adapted to benefit from the minimal-height tree diagnostic generation algorithm implemented in the CÆSAR_SOLVE library. In some cases, the size of counterexamples produced by depth-first search resolution algorithms was reduced by nearly two orders of magnitude, which makes them suitable for visual inspection.
- In order to validate BISIMULATOR and compare its performance with other equivalence checking tools, extensive experiments were performed on 70 realistic examples obtained from the CADP distribution and the VLTS benchmark suite (see § 6.3.3). The results indicate that BISIMULATOR compares favorably with the on the fly algorithms implemented in ALDÉBARAN. In some cases (e.g., for observational equivalence), BISIMULATOR even outperforms the classical algorithms based on partition refinement.

6.1.4. The EVALUATOR 4.0 Tool

Participant: Radu Mateescu.

EVALUATOR 4.0 is an on the fly model checker for temporal properties containing data. These properties are evaluated on a graph represented implicitly using the OPEN/CÆSAR environment. The input language of EVALUATOR 4.0 is the regular alternation-free μ-calculus extended with typed variables. It offers primitives for handling states and transitions in logic formulas, thus allowing to express non-standard properties (such as for instance the fact that a state has a looping transition on itself, or that a finite sequence of transitions has the same number of occurrences of two actions A and B).

In 2003, we continued the development of EVALUATOR 4.0. Our work focused on improving the translation of the verification problem into a boolean equation system. We devised two property-preserving reductions, which are performed on the fly on the OPEN/CÆSAR graph:

- *τ-compression* merges all states of a strongly connected component that contains only τ-transitions. This reduction preserves the usual weak equivalences (e.g., branching, observational, $\tau^*.a$, and safety equivalences) and, thus, any property compatible with these equivalences. The algorithm is derived from Tarjan's algorithm for computing strongly connected components.
- *τ-closure* collapses sequences of τ-transitions preceding each visible transition. This reduction preserves $\tau^*.a$ and safety equivalences and, thus, any property compatible with these equivalences (such as the safety property "[R] *false*", which forbids any execution sequence matching the regular expression R). The algorithm computes a transitive closure using two nested depth-first searches.

These reductions significantly increase the speed of EVALUATOR 4.0 (by a factor up to 5 for graphs containing many τ-transitions).

6.1.5. The AAL Tool

Participants: Alban Catry, David Champelovier, Hubert Garavel, Radu Mateescu.

In the framework of the ARCHWARE project (see § 7.1), we focused on the analysis of software architectures:

- We contributed to the definition of AAL (*Architecture Analysis Language*) [26], a language for expressing properties of software architectures and architectural styles. 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).
- We defined AAF-MC (*Architecture Analysis Formalism for Model Checking*) [27], the fragment of AAL containing properties to be verified using model checking. In collaboration with ARCHWARE partners, we identified a large number of property patterns relevant to software architectures. We also studied fragments of AAF-MC compatible with usual equivalence relations (e.g., strong, branching, observational, and safety equivalences).
- We developed a prototype model checker for AAF-MC [28], which translates the temporal formulas expressed in AAF-MC into boolean equation systems. This prototype tool ($7,500$ lines of code) was developed using the SYNTAX compiler generator and the TRAIAN compiler (see § 5.2).
- We also developed a methodology for the efficient verification of AAF-MC properties on execution traces generated during the simulation of an architectural description [30].

6.1.6. Partial Order Reduction Tools

Participants: Frédéric Lang, Radu Mateescu.

A way to fight state explosion is partial order reduction, which tries to avoid the exploration of redundant interleavings caused by independent, concurrent transitions. A form of partial order reduction that preserves branching equivalence is the so-called τ*-confluence* [43]. It consists in identifying, for each state s, the set $C(s)$ of *confluent* τ-transitions going out of s, meaning that after their execution any other transition going out of s can still be executed. The reduction consists in eliminating all transitions going out of s except those in $C(s)$.

In 2003, we developed a τ-confluence reduction tool (800 lines of C code) for graphs represented implicitly using the OPEN/CÆSAR environment. This tool encodes the definition of τ-confluence as a boolean equation system, which is solved on the fly using the general algorithms A1 and A2 provided by the CÆSAR_SOLVE library (see § 6.1.2). Experiments on various communication protocols indicate that τ-confluence significantly reduces the number of states (up to 6 times) and transitions (up to 10 times).

The work on τ-confluence led to a publication [24].

6.1.7. The EXP.OPEN 2.0 Tool

Participants: Frédéric Lang.

EXP.OPEN 2.0 is a compositional verification 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), or "graphical" operators (as in E-LOTOS and LOTOS NT [10]), 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.

Version 2.0 of EXP.OPEN was developed in 2002 to overcome the limitations of the previous version 1.0. In 2003, we worked along the following lines:

- We proved that branching bisimulation is a congruence for all EXP.OPEN 2.0 operators. This is a key property for compositional verification.

- The code was optimized. A performance comparison with EXP.OPEN 1.0 on a dozen realistic examples shows that EXP.OPEN 2.0 is faster (from 1.6 to 50 times) and uses less memory (about 2 times less).
- We extended the possibilities of translation from EXP.OPEN's networks of communicating automata into input formats accepted by other tools. In addition to the existing interconnection with the PEP tool for Petri nets (developed at the University of Oldenburg), we implemented a translation to the parallel FC2 format designed at INRIA Sophia-Antipolis.
- To address state explosion, we enhanced EXP.OPEN with a partial order reduction technique based on τ-confluence (see § 6.1.6). We assessed our approach on two benchmarks (a leader election protocol on a token ring, and a distributed implementation of Erathostene's sieve) used by Ramakrishna & Smolka [49] to experiment another partial order reduction based on τ-*inertness*. The graphs generated by EXP.OPEN are comparable or even smaller than those reported in [49]: As the number of concurrent processes increases, the graph generated by EXP.OPEN for the sieve remains of constant size, whereas [49] indicates a linear growth.

The EXP.OPEN 2.0 tool consists of $1,700$ lines of SYNTAX code, $6,400$ lines of LOTOS NT code, and $1,600$ lines of C code. An article about EXP.OPEN 2.0 was submitted to an international conference.

6.1.8. Parallel and Distributed Verification Tools

Participants: Nicolas Descoubes, 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.

Our initial efforts focused on parallelizing the graph construction algorithm [7], which is a bottleneck for verification as it requires a considerable memory space to store all reachable states. In this respect, we developed the following software:

- DISTRIBUTOR 2.1 splits the construction of a graph over N machines communicating using sockets. Each machine is required to build a fragment of the graph represented as a BCG file, the states being distributed between the N machines by means of a statically determined hash function.
- BCG_MERGE 3.0 merges the N graph fragments constructed by DISTRIBUTOR to obtain — after renumbering states appropriately — a unique BCG file representing the entire graph.
- CÆSAR_NETWORK is a code library for distributed tools such as DISTRIBUTOR and BCG_MERGE. It provides basic functionalities including: management of the machine configuration file that contains the parameters of the distributed computation, process deployment protocol on a set of remote machines, emission and reception of messages using blocking or non-blocking sockets, communication buffer management, etc. CÆSAR_NETWORK allows a clear separation between verification algorithms and communication primitives.

In 2003, we improved the distributed model checking tools as follows:

- We ported the CÆSAR_NETWORK library to the WINDOWS and LINUX operating systems.
- We integrated new functionalities in the CÆSAR_NETWORK library, such as a better error processing, primitives to enable both blocking and non-blocking message sending in the same algorithm, and several bug corrections.
- We extended DISTRIBUTOR to handle graphs containing labels of unbounded length, which requires a non-trivial distributed algorithm.

- DISTRIBUTOR was made faster (from 2 to 5 times depending on the size of graphs and number of machines) by reducing the time spent in busy waiting.
- BCG_MERGE was made (at least one order of magnitude) faster by delegating to remote machines certain tasks (such as opening, closing, and sorting BCG files) that were previously done sequentially by the coordinating process.
- We developed a prototype, distributed version of the CÆSAR_SOLVE (see § 6.1.2) generic library. This prototype library (7,600 lines of C code) uses several machines to solve boolean equation systems on the fly. It implements an efficient parallel version of algorithm A2 based on a breadth-first traversal of the boolean graph.

An article on distributed model checking was published [21].

6.1.9. Other Tool Developments

Participants: Damien Bergamini, Aurore Collomb, Nicolas Descoubes, Hubert Garavel, Christophe Joubert, Frédéric Lang, Frédéric Tronel.

We also improved the following CADP tools and libraries:

- The CÆSAR_HASH library was enriched with several new hash functions.
- The BCG_IO tool was enhanced with a new option that produces the graph format accepted by the ETMCC (*Erlangen-Twente Markov Chain Checker*) tool.
- The BCG_INFO tool was enhanced with a new option that reports about the order (depth-first search, breadth-first search, etc.) in which the transitions of a graph are sorted. Also, the algorithm used by BCG_INFO to decide whether a graph is deterministic was redesigned to avoid loading the entire transition relation in main memory, thus resulting in significant time savings (from several hours to a few minutes for large graphs).
- The DETERMINATOR, BCG_STEADY, and BCG_TRANSIENT tools were ported to the WINDOWS operating system. BCG_STEADY and BCG_TRANSIENT were enhanced so as to generate output in the standard CSV (*Comma Separated Values*) format used by mainstream data processing tools, including EXCEL and GNUPLOT. Two demo examples illustrating the use of CADP for performance evaluation were prepared. A tool paper on DETERMINATOR, BCG_STEADY, and BCG_TRANSIENT was published [20].
- The SVL language was extended to generate particular graphs using the BCG_GRAPH tool (see § 6.1.1), to minimize stochastic and probabilistic BCG graphs using the BCG_MIN tool, and to support shell variables representing lists of labels instead of single labels.
- Finally, we started to adapt the CADP tools to the latest versions of operating systems (LINUX REDHAT 9, etc.) and compilers (GCC 3), and to improve their integration with WINDOWS (especially by supporting filenames containing space characters).

We also contributed to other tools outside the VASY team:

- We corrected several bugs in the TREX symbolic verification tool[1] and added several functionalities, such as gate synchronization, optimized representation of boolean variables, and new options for controlling acceleration and processing non-linear formulas. These improvements led to the release of TREX 1.3 in February 2003 and TREX 1.4 in November 2003.
- We found and proposed a correction for a bug in the SYNTAX abstract tree generator developed at INRIA Rocquencourt (incorrect error message when source code is split in several files).

[1] See http://www.liafa.jussieu.fr/~sighirea/trex

6.2. Languages and Compilation Techniques

6.2.1. Compilation of LOTOS

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

In 2003, work took place — essentially in the framework of the FORMALFAME contract (see § 6.3.1 and § 7.2) — to enhance the LOTOS tools present in the CADP toolbox. As regards the CÆSAR.ADT compiler for the data part of LOTOS:

- We fixed 4 remaining bugs in the compiler front-end common to CÆSAR and CÆSAR.ADT (lexical analysis, syntactic error recovery, and static semantics).
- We designed and implemented a fixpoint algorithm for the detection of LOTOS types whose domain of values is either finite or manually bounded by the specifier.

As regards the CÆSAR compiler for the process part of LOTOS:

- We fixed 2 bugs in CÆSAR's optimization E2 and simulation phases.
- The EXEC/CÆSAR programming interface was enhanced with new functionalities that give access to Petri net-related information (number of places, number of transitions, last transition fired, etc.), as well as means to randomize the firing of τ-transitions; these features proved to be useful for the FORMALFAME contract.
- We introduced a heuristical algorithm that permutes the various fields of state vectors so as to save memory by reducing the unused "padding" bits introduced by machine word alignment constraints; although the average memory gain measured on a large set of benchmarks is disappointing (2%), it is still worth when millions of state vectors are to be stored in main memory.
- Similarly, to reduce the memory size of transition labels, we introduced several optimization techniques: compaction of transition numbers, use of bit fields, and field permutation (as for state vectors), all of which led to an average memory gain of 45%.
- The code generated by CÆSAR for computing a hash function on transition labels was improved; in practice, the average number of hash collisions is divided by a factor ranging from 2.5 (on PC/LINUX) to 3.9 (on SUN/SOLARIS).
- The code generated by CÆSAR for converting transition labels to character strings was improved in several ways; in practice, this makes the exploration of an entire transition system (between 1.25 and 4 times) faster.
- In the EXEC/CÆSAR mode, the code generated for firing each transition was made faster by delaying as much as possible certain computations (e.g., current state storage and next state computation) until it is sure that the transition will be actually fired (which is only the case if all guard predicates are true and if the environment selects that particular transition); otherwise, these computations can be safely skipped. Another profitable optimization consists in avoiding to recompute the successor state information several times for the same state (a situation that may occur when the environment is not immediately ready to accept a transition). On the ILU benchmark studied in the FORMALFAME contract, these optimizations led to a speed improvement between 24% (on PC/LINUX) and 43% (on SUN/SOLARIS).

Additionally, we also investigated techniques for state space reduction, our goal being to decrease the size of the graphs generated by CÆSAR, still preserving strong bisimulation between the original and reduced graphs.

We considered the approach based on live variable analysis, first proposed by H. Garavel and Juan Galvez [53]. The basic idea is to assign a canonical value to any variable that is no longer used, so as to avoid distinguishing state vectors that only differ by the values of some variables not used in the future. This is done by adapting classical data flow analysis to the extended Petri nets generated by CÆSAR and by resetting to zero each variable as soon as it ceases to be alive.

In 2003, we generalized the approach of [53] to handle so-called *hierarchical units*, i.e., the possibility to split each process into a set of concurrent sub-processes at an arbitrary nesting depth. In this model, concurrent processes do not share variables; however, the variables of a parent process can be consulted (but not modified) by its children sub-processes, a situation for which we designed several heuristics.

We implemented our ideas in a prototype version of CÆSAR (about $3,600$ lines of additional C code), which we applied to a benchmark suite of 469 LOTOS specifications. For 98 examples (21%), the size of graphs generated by CÆSAR was divided by a mean factor of 11.6. On some examples, we even observed a reduction factor of 300.

6.2.2. Compilation of the E-LOTOS Data Part

Participants: David Champelovier, Hubert Garavel.

As regards the data part of E-LOTOS, we continued to improve the TRAIAN compiler (see § 5.2), which is distributed on the Internet (see § 9.1) and used intensively within the VASY team as a development tool for compiler construction [6].

In 2003, we released a new version 2.3 of TRAIAN, which supersedes the previous version 2.2 issued in 2002. This development effort, which increased the software size from $48,000$ to $55,000$ lines of code, completes the integration in TRAIAN of the code optimizations studied by Claude Chaudet in 1999 (see § 5.2.3 in the 1999 VASY activity report and § 5.2.1 in the 2002 VASY activity report). It also brings a higher degree of symmetry between TRAIAN and the CÆSAR.ADT compiler for the data part of LOTOS (see § 5.1). In addition to several bug fixes, the new version of TRAIAN brings useful enhancements:

- Particular classes of LOTOS NT data types (enumerated types, tuples, natural numbers, singleton types, and isomorphic types) are now recognized automatically and implemented optimally.
- For recursive types, heuristics allow to reduce the number of LOTOS NT types implemented using pointers; however, a compiler directive exists to force a given LOTOS NT type to be implemented using pointers.
- It is now possible to give pointer types a canonical representation by storing all their values into hash tables, which avoids to allocate multiple instances of the same value; in the framework of enumerative verification, this technique allows significant savings in memory space (on a benchmark proposed by Jan Friso Groote, we observed that the amount of memory needed was divided by 400).
- As for LOTOS, it is now possible to split LOTOS NT specifications into several files using a compiler directive.

In parallel, we pursued the design of TRAIAN 3.0, a new generation compiler that could handle the data parts of both LOTOS and LOTOS NT, so as to merge CÆSAR.ADT and TRAIAN 2.3 into a unique compiler. In 2003, the requirement base for TRAIAN 3.0 grew from 140 to 198 entries.

6.2.3. Compilation of the E-LOTOS Process Part

Participants: Aurore Collomb, Hubert Garavel, Frédéric Lang, Guillaume Schaeffer.

Compiling the process part of E-LOTOS and LOTOS NT is a difficult problem as these languages combine concurrency, quantitative time, and exceptions. To deal with these problems progressively, we chose to focus first on the sequential processes present in E-LOTOS and LOTOS NT. We designed a formalism named NTIF (*New Technology Intermediate Form*) to be used as an intermediate language for compiling and verifying E-LOTOS and LOTOS NT processes.

NTIF allows to specify extended automata parameterized by typed variables. Each transition is labeled with an action (which allows communication with the environment according to the rendezvous semantics of process algebras) and a sequential code fragment to read and/or write variables. Compared to classical "condition/action" (or "guarded commands") automata, NTIF provides high level control structures (statements "*case*", "*if-then-else*", "*while*", etc.); this avoids the introduction of spurious intermediate states and transitions, as well as the duplication of boolean conditions, an important source of errors [5].

In 2003, we started introducing quantitative time concepts in NTIF. In the vein of E-LOTOS, we added a "*wait*" operator that lets a given amount of time elapse, timing tags on actions to express deadline and urgency, and a construct to capture the time elapsed between the instant an action is enabled and the instant it actually occurs. We defined the semantics of this extension and started to demonstrate suitable properties (e.g., time additivity) using the COQ theorem prover. The semantics was also assessed by modeling several classical timed protocols (e.g., Bounded Retransmission Protocol, Fisher protocol, etc.) using NTIF.

In parallel, the existing tools for NTIF were enhanced in several ways:

- For modularity reasons, we merged the NT2DOT (which visualizes NTIF descriptions graphically) and NT2IF (which unfolds NTIF descriptions to produce lower level formalisms) into one single tool, named NTIF. The architecture of this new tool supports several compiler back-ends that translate NTIF into a variety of languages and formats.
- A file inclusion mechanism was implemented, which allows to split NTIF descriptions into several files.
- The static semantics checking phase of NTIF was entirely rewritten to be more efficient and display better error messages. Static checks for proper variable initialization were added, which allowed to detect uninitialized variables in existing NTIF specifications.
- Two new back-ends were developed, which translate NTIF to the input languages used by the TREX and UPPAAL tools for timed automata. In the case of UPPAAL, both XTA and XML formats can be generated. The back-ends translate the high level NTIF timed constructs into clocks, time guards, and time progress conditions that express the impossibility to enter or stay in a given state.
- We added to NTIF two standard libraries (lossy buffers and write-only buffers). In the general case, these buffers are expressed as normal NTIF processes. However, when translating to TREX input language, these buffers are recognized and implemented as TREX built-in buffers for optimization purpose.

These improvements increased the size of the NTIF tool from $6,500$ to $13,300$ lines of code ($9,500$ lines of LOTOS NT code, $2,200$ lines of SYNTAX code, and $1,600$ lines of C code).

6.3. Case Studies and Practical Applications

6.3.1. The FAME Cache Coherency Protocol

Participants: Damien Bergamini, Hubert Garavel, Radu Mateescu, Solofo Ramangalahy.

Since October 1998, we have been co-operating with BULL in the framework of the FORMALFAME contract (see § 7.2) devoted to the use of formal methods for verifying and testing multiprocessor architectures. Our work targets at FAME, the CC-NUMA architecture developed by BULL for its NOVASCALE series of high-performance servers based on 64 bits INTEL ITANIUM processors. The core of FAME is a complex circuit (named B-SPS), which implements the FAME cache coherency protocol and performs routing between processors and input/output nodes. The complexity of this circuit comes both from the internal parallelism inherent to routing and from multiple accesses to shared data, the consistency of which must be preserved.

After studying the B-SPS as a whole (1999–2002), FORMALFAME currently focuses on two complex subcomponents of B-SPS: the PRR (*Pending Request Responses*) block and the ILU (*InterLeaving*) unit, which are modeled and analyzed in great detail. In 2003, work took place in two directions:

- *Formal specification:* The development of LOTOS specifications for PRR (about $7,500$ lines of LOTOS and 200 lines of C code) and ILU (about $8,900$ lines of LOTOS and $3,400$ lines of C code) was done at BULL Les Clayes-sous-Bois by Yehong Xing and Jack Abily. Two successive versions

V1 and V2 were described, the former corresponding to a chip sent to foundry in March 2003 and the latter to the forthcoming evolution of this chip. The role of VASY was essentially to provide expertise in writing LOTOS specifications (for instance, on the proper way of introducing random number generation in formal models, on the decision to implement a given feature either as a LOTOS process or as a LOTOS data operation, etc.).

- *Validation of* VERILOG *designs:* A long-standing goal of FORMALFAME is to reuse the LOTOS specification developed for a hardware system (or unit, or block, respectively) as a means to check the correctness of the VERILOG design implementing this system (or unit, or block). So doing, the effort invested in formal specification is rewarded not only by finding mistakes at a high level of abstraction, but also in the actual system implementation itself.

  In 2003, we experimented two different approaches to validate the VERILOG designs of PRR and ILU.

  As regards PRR, the conformance between the LOTOS specification and the VERILOG design is checked by means of random test sequences generated automatically from the LOTOS specification (using the EXECUTOR tool of CADP). This approach is justified by the deterministic nature of PRR, which always emits the same outputs when given the same inputs. The generated test sequences contain both input and output events; they are applied to the VERILOG design of PRR using CADENCE's TESTBUILDER tool [35]. This approach proved to be successful as large test sequences (more than $100,000$ transactions) were produced and allowed to find a problem in PRR V1 (technically, a non-conformity between the LOTOS specification and the VERILOGdesign). This problem was not detected by other techniques because it has no impact on global cache coherence/memory consistency.

  As regards ILU, the conformance between the LOTOS specification and the VERILOG design is checked by translating the LOTOS specification into a C program that is connected to the VERILOG design of ILU using the EXEC/CÆSAR and TESTBUILDER tools. This C program drives the execution of the VERILOG design, sending inputs and checking the correctness of received outputs. The inputs are generated randomly using an history-based guidance mechanism, which gives priority to branches that have not been chosen already. The approach targets at the extensive testing of the VERILOG design by letting this guided simulation execute automatically for a certain amount of time until a suitable coverage level is reached. For ILU, the coverage criterion used in 2003 is the percentage of transitions (in the Petri net generated by CÆSAR from the LOTOS specification of ILU) fired during the guided simulation. This approach proved to be technically challenging due to the intrinsic complexity of the ILU behavior and to the programming machinery needed to interface the ILU design with its formal specification. For the approach to be effective, two main issues had to be solved: The simulation speed was initially too slow (about 360 visible transitions per second) and the coverage was originally too small (15% of transitions fired). These issues were addressed in several ways:

  - The LOTOS specification of ILU was modified to define various hardware data structures (e.g., associative search tables, directories, etc.) using data types and functions rather than LOTOS processes.
  - The efficiency of the C code generated by CÆSAR was significantly improved (see § 6.2.1).
  - The firing of τ-transitions evolved from fixed priority order to random order.

  These changes led to a satisfactory solution (speed multiplied by a factor of 3 and coverage increased up to nearly 100%). The FORMALFAME approach is now fully operational and used intensively for the validation of PRR V2 and ILU V2. The next research steps of FORMALFAME investigate a different coverage criterion: the percentage of reachable, visible labels produced during the guided simulation (thus measuring coverage for both LOTOS data types and gate offers).

6.3.2. The SCALAGENT Deployment Protocol

Participants: Hubert Garavel, Frédéric Lang, Radu Mateescu, Frédéric Tronel.

In the framework of the PARFUMS contract (see § 7.3), we studied a deployment protocol designed and implemented by SCALAGENT, a startup company originating from the former SIRAC team of INRIA Rhône-Alpes. Among many potential applications, this protocol can be used to install and configure a set of mobile agents (JAVA components) on the uninterruptible power supplies manufactured by the MGE-UPS company.

The approach followed for the verification of the deployment protocol consists in generating automatically a formal specification in LOTOS for a given configuration of agents to be deployed. The automatic generation is accomplished by a translator that, given as input an XML file produced by a graphical configuration editor, produces a set of corresponding LOTOS processes and an SVL script that drives the verification. The translator takes into account the JAVA class hierarchy and inheritance relationships to synthesize the dynamic behavior of each agent.

Given the high complexity of the deployment protocol (which features a set of distributed agents, each agent consisting itself of a set of concurrent activities), we chose a compositional verification approach reflecting closely the deployment protocol architecture. Virtually, all tools of CADP are used during the verification process, including the BCG_GRAPH tool (see § 6.1.1) especially developed to generate communication buffers efficiently.

In 2003, we completed the deployment protocol verification, with the following results:

- Model checking verification revealed some obscure points in the protocol specification, which we clarified in accordance with the SCALAGENT designers. In particular, the original specification was silent about the model of communications between activities. In this respect, verification made clear (by exhibiting an infinite loop of error messages) that the protocol was not meant for handling asynchronous messages between inner activities and would only function properly with communications implemented as local procedure calls (i.e., the calling activity gets suspended until the procedure call returns).
- We applied the translator to configurations of increasing sizes, up to 71 concurrent processes. Although the high degree of concurrency could have lead to state explosion (potentially $9 \cdot 10^{68}$ states), compositional verification allowed to keep the size of intermediate state spaces under a workable limit (below 10^6 states).

These results led to a publication [25].

6.3.3. The VLTS Benchmark Suite

Participants: Damien Bergamini, Nicolas Descoubes, Hubert Garavel.

In collaboration with the SEN2 team of CWI (Amsterdam, The Netherlands), we developed and made available to the scientific community the VLTS benchmark suite (see http://www.inrialpes.fr/vasy/cadp/resources/benchmark_bcg.html).

First benchmark base of this kind, VLTS (*Very Large Transition Systems*) is a collection of forty labeled transition systems of increasing sizes (ranging from 300 states to 34 million states). It provides a scientific criterion for a performance assessment of algorithms and tools operating on large graphs, including graph visualization software, explicit state verification software (model checkers, equivalence checkers, and minimization tools), as well as computer formats for the representation of transition systems. Although very recent, the VLTS benchmark suite has already been used in several scientific publications.

The benchmarks of the VLTS are encoded in the BCG format and benefit from the specific compression techniques provided by this format. The development of the VLTS benchmark base led us to a finer tuning of the default compression parameters of the BCG format, resulting in a better compression (approximately 12%). Moreover, we developed additional compression techniques and tools, which decreased the amount of the disk space needed to store the entire VLTS from 800 down to 454 Mbytes.

6.3.4. Other Case Studies

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

In 2003, the VASY team also worked on the following case studies:

- We verified compositionally (using SVL scripts) a randomized consensus protocol [36][48], in which several distributed processes try to agree on a common value. The convergence is guaranteed, but only after an unbounded number of rounds. We generated the BCG graph for one round and showed that the BCG graph for several rounds can be obtained by induction from the former graph using label renaming and parallel composition only. This case study allowed us to experiment the new BCG_GRAPH tool (see § 6.1.1) to generate the communication media.
- We reconsidered a former case study: the verification of a cache memory (originally described in [33]), which was verified using abstractions and CADP as described in [42]. While some years ago the verification of the abstracted protocol was only feasible using sophisticated compositional techniques, the latest version of CÆSAR managed to generate a smaller graph (about $10,000$ states) directly.
- We extended our NTIF model of the CEPS (*Common Electronic Purse Specification*, see § 5.2.1 in the 2001 VASY activity report) with new functionalities, such as for instance the "*Load device*" operation. This extended model ($1,300$ lines of NTIF code, 102 variables among which 11 symbolic parameters) was used as a common case study for several verification and testing tools (TREX, HARVEY, ATELIER B, STG, and BZ-TT). The VASY team carried out the verification using TREX, which generated the graph of reachable symbolic states in 1h42. Some properties, such as "*all slots used contain different currencies*", were verified in a few seconds using EVALUATOR 3.0 (see § 6.1.4).
- In collaboration with Carmen Occhipinti and Paolo Fabriani (Engineering, Italy), we specified in LOTOS the FKMS (*Federated Knowledge Management System*) case study proposed by Engineering in the framework of the ARCHWARE project (see § 7.1). Two different versions of the LOTOS specification were developed, containing a simple federation manager (750 lines of code) and an enhanced one ($1,100$ lines of code) [29]. We then expressed several temporal logic properties characterizing the proper handling of documents within the FKMS and checked them on several configurations using EVALUATOR 3.0. This activity allowed to spot an error in the querying mechanism of documents in an FKMS ontology.
- In collaboration with Nathalie Chabrier, François Fages, and Nathalie Sznajder (CONTRAINTES team of INRIA Rocquencourt), we investigated the use of CADP for analyzing formal models of biological processes expressed in the BIOCHAM language developed by CONTRAINTES. A translator from BIOCHAM to LOTOS was developed and applied to a qualitative model of cell cycle control for mammals; on the generated LOTOS code, it was then possible to verify various temporal properties using the EVALUATOR 3.0 model checker. This experiment led to several findings. The syntax and static semantics rules of LOTOS unveiled various mistakes, which had not been caught before; this problem was addressed by CONTRAINTES in its revised version of BIOCHAM. Also, it appeared that the LOTOS code generated by the translator was excessively verbose; for this reason, VASY started to incorporate in its LOTOS compilers some useful shorthand notations that exist in E-LOTOS and LOTOS NT.
- In collaboration with Grégory Batt and Hidde de Jong (HELIX team of INRIA Rhône-Alpes), we connected 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. To achieve the connection, we developed a translator named GNA2BCG (200 lines of C code) from

the graph format produced by GNA to the BCG format. The resulting BCG graph can be simplified by eliminating instantaneous states using abstraction and reduction modulo branching equivalence, and inspected visually by using the BCG_EDIT tool of CADP. Then, various temporal properties concerning the behavior of the regulatory network (evolution of protein concentrations, reachability of equilibrium states, etc.) can be verified using the EVALUATOR 3.0 model checker. In addition, the input language of EVALUATOR 3.0 was extended with an operator for string concatenation, which allows a better parameterization of property definitions, yielding more concise specifications. This verification methodology was applied to the analysis of a small biologically-inspired system.

Other teams also used the CADP toolbox for various case studies. To cite only the work published in 2003, we can mention:

- the synthesis and verification of constraints for the Pragmatic General Multicast (PGM) protocol [38],
- the verification of the cache coherency protocol of the JACKAL distributed shared memory implementation of JAVA [47],
- the verification of an agent-oriented auction protocol [40],
- the verification of a distributed fault tolerant algorithm developed using the JAVASPACES coordination architecture [54],
- the architectural unit testing of a robot teleoperation system described in UML [50],
- the verification of distributed dataspace applications developed using the *space calculus* [46],
- the verification of the Positive Acknowledgement Retransmission (PAR) protocol [37].

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 VODKAV tool, which uses CADP to extract information from a system described in ERLANG, with application to a video-on-demand server [34],
- a translator of SoC (*System on Chip*) descriptions based on the STBus developed by STMicroelectronics into LOTOS for codesign and verification purposes [51],
- the ELSE state space generator, which explores the symbolic reachability graph of timed automata in connection with CADP [52].

7. Contracts and Grants with Industry

7.1. The IST ArchWare European Contract

Participants: Damien Bergamini, Alban Catry, David Champelovier, Aurore Collomb, Nicolas Descoubes, Hubert Garavel, Christophe Joubert, Frédéric Lang, Radu Mateescu.

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 (LLP/CESALP 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. Based on a software architecture description language, this environment will offer functionalities to define architectural styles specific to various activity domains, as well as engineering tools for analyzing architectural descriptions. The role of VASY in ARCHWARE concerns the description and verification of functional properties.

In 2003, we contributed to the definition of AAL (*Architecture Analysis Language*) for expressing properties of software architectures, to the development of a prototype model checker for the AAL fragment dedicated to behavioral properties (see § 6.1.5), and to the formal specification of Engineering's FKMS (*Federated Knowledge Management System*) (see § 6.3.4).

7.2. The FormalFame Contract

Participants: Damien Bergamini, Hubert Garavel, Radu Mateescu, Solofo Ramangalahy.

Since 1995, there has been a long-standing collaboration of VASY and BULL, to which the former PAMPA team of INRIA Rennes participated until December 2000. This collaboration aims at demonstrating that the formal methods and tools developed at INRIA for validating and testing telecommunication protocols can also be successfully applied to BULL's multiprocessor architectures. The long-term objective is to develop a complete and integrated solution supporting formal specification, simulation, rapid prototyping, verification, test generation, and test execution.

A first phase of this collaboration took place from 1995 to 1998 in the framework of the DYADE joint venture between BULL and INRIA. Two case studies were successfully tackled: the POWERSCALE bus arbitration protocol [39] and the POLYKID multiprocessor architecture [11]. The feasibility of the proposed approach was established and BULL expressed its interest in pursuing the collaboration for its new architectures.

Since October 1998, we have been working on FAME, the CC-NUMA multiprocessor architecture developed by BULL for its NOVASCALE series of high-performance servers based on INTEL ITANIUM 64 bits processors. Initially informal, this collaboration was officialized in 1999 as a DYADE action named FORMALFAME, which lasted until the end of DYADE in March 2001. Since then, the collaboration is going on under the form of a BULL-INRIA contract, for which we kept the name FORMALFAME. This contract gathers the VASY team and the BULL team of Sylvie Lesmanne (who succeeded Anne Kaszynski in March 2002) in Les Clayes-sous-Bois, the coordination being ensured by a BULL engineer (M. Zendri, then N. Zuanon, then S. Ramangalahy) located at INRIA Rhône-Alpes.

FORMALFAME successively focused on several critical components of the FAME architecture: the CCS circuit that manages communications for a group of four processors and the NCS circuit that manages network communications (from October 1998 to November 1999), the B-SPS circuit that implements the cache coherency protocol (from December 1999 to March 2002), and, since then, the PRR block and the ILU unit, which are two sub-components of the B-SPS circuit to which a particular attention is drawn. For each of these components, LOTOS descriptions were written, which provided a formal basis for testing and verification.

In 2003, the results obtained by FORMALFAME concern, on the one hand, the validation and testing of PRR and ILU (see § 6.3.1) and, on the other hand, the improvement of various CADP tools.

7.3. The RNTL Parfums Contract

Participants: Hubert Garavel, Frédéric Lang, Radu Mateescu, Frédéric Tronel.

PARFUMS (*Pervasive Agents for Reliable and Flexible UPS Management Systems*) is a pre-competitive RNTL (*Réseau National des Technologies Logicielles*) two year project. Started in May 2001, PARFUMS gathers three companies (MGE-UPS, SCALAGENT — a startup originating from the former SIRAC team of INRIA, and SILICOMP) and the VASY team of INRIA.

PARFUMS aims at implementing a safe, flexible architecture (based on JAVA components) that enables the remote management of uninterruptible power supplies from various equipments (computers, mobile phones, personal assistants, etc.). This architecture relies on the SCALAGENT software infrastructure facilities (fault tolerant software bus, mobile agents communicating by messages, and associated development tools) in order to ease the deployment, supervision, reconfiguration, and update of administrative software, while minimizing the cost of these operations.

In 2003, the contribution of VASY to the PARFUMS project focused on the modeling and compositional verification of the SCALAGENT deployment protocol (see § 6.3.2).

8. Other Grants and Activities

8.1. National Collaborations

In 2003, we collaborated with several INRIA teams:

- APACHE (Rhône-Alpes): use of the "i-cluster" computer to experiment parallel and distributed verification algorithms (see § 6.1.8);
- CONTRAINTES (Rocquencourt): applications of model checking to biochemical systems (Nathalie Chabrier, François Fages, and Nathalie Sznajder);
- HELIX (Rhône-Alpes): applications of model checking to biological systems (Grégory Batt and Hidde de Jong);
- LANDES (Rennes), LEMME (Sophia-Antipolis), OASIS (Sophia-Antipolis) and VERTECS (Rennes): collaboration in the framework of the MODOCOP project.

Beyond INRIA, we have scientific relations with the following teams:

- ISIMA/LIMOS laboratory (Clermont-Ferrand): methods for hardware-software co-design combining LOTOS, LOTOS NT, and VHDL (Pierre Wodey);
- LAAS-CNRS laboratory (Toulouse): interconnection of the TINA tool of LAAS-CNRS with the CADP toolbox (Bernard Berthomieu and Francois Vernadat);
- LIAFA laboratory (Paris): development of the TREX symbolic verification tool (Mihaela Sighireanu);
- LIFC laboratory (Besançon): verification of an electronic purse smartcard applet modeled in NTIF and translated into B (Fabrice Bouquet and Alain Giorgetti).

8.2. International Collaborations

8.2.1. International Working Groups

- 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.
- H. Garavel is a member of the technical committee (*ETItorial Board*) of the ETI (*Electronic Tool Integration*) software development platform.

8.2.2. Bilateral International Relations

We maintain scientific relations with several international universities and research centers. In addition to our partners in aforementioned contractual collaborations, we had scientific exchanges in 2003 with:

- CWI (Stefan Blom, Wan Fokkink, and Jaco van de Pol),
- Université Libre de Bruxelles (Thierry Massart),
- University of Göthenburg in Sweden (Thomas Arts),
- University of Kent at Canterbury (Clara Benac Earle),
- University of Saarbrücken (Holger Hermanns),
- University of Twente (Axel Belinfante).

8.3. Visits and Invitations

- Carmen Occhipinti and Paolo Fabriani, from Engineering (Rome), visited us on February 4–5, 2003.
- Fabien Leymonerie, David Le Berre, and Lionel Blanc, from Ecole Supérieure d'Ingénieurs en Informatique d'Annecy, visited us on February 14, 2003, in the framework of the ARCHWARE European project.
- Mihaela Sighireanu, from the LIAFA laboratory, University of Paris 7, visited us on July 3, 2003. She gave a talk entitled "*Synthesis and verification of constraints in the PGM protocol*".
- Dominique Borrione, Menouer Boubekeur, and Emil Dumitrescu, from the TIMA laboratory (Grenoble), Francois Fages and Nathalie Chabrier, from the CONTRAINTES team of INRIA Rocquencourt, and Jean-Pierre Gallois, Christophe Gaston, and Nicolas Rapin, from CEA (Saclay), participated in the annual VASY seminar, held in Saint-Pierre de Chartreuse on June 10–13, 2003:

  – D. Borrione gave a talk entitled "*Présentation des travaux du groupe VDS sur la vérification de systèmes matériels*",
  – M. Boubekeur gave a talk entitled "*Expérimentation de CADP pour la validation de spécifications de circuits asynchrones écrites en CHP*",
  – E. Dumitrescu gave a talk entitled "*La simulation symbolique comme stratégie de simplification pour la vérification de systèmes sur la puce par model checking*",
  – F. Fages gave a talk entitled "*Model checking symbolique de réseaux biochimiques*",
  – N. Chabrier made a demonstration of BIOCHAM (*Biochemical Abstract Machine*),
  – J.-P. Gallois, C. Gaston, and N. Rapin gave a talk entitled "*Analyse comportementale de spécifications formelles à base d'automates par dépliage*" and made a demonstration of the AGATHA tool.

9. Dissemination

9.1. Software and Internet Dissemination

The VASY team distributes two main software tools: the CADP toolbox (see § 5.1) and the TRAIAN compiler (see § 5.2). In 2003, the main facts are the following:

- We prepared and distributed successive beta-versions (2002-g, ..., 2002-w) of CADP.
- The number of license contracts signed for CADP increased from 276 to 307.
- We were requested to grant CADP licenses for 838 different computers in the world.
- The distribution of the TRAIAN compiler continued and a new version 2.3 of TRAIAN (see § 6.2.2) was released on May 5, 2003.
- The TRAIAN compiler was downloaded by 98 different sites.

The VASY WEB site was regularly updated with scientific contents, announcements, publications, etc. Most notably, the following contents were produced in 2003:

- The CADP *Frequently Asked Questions* page was enriched with 30 new articles (see http://www.inrialpes.fr/vasy/cadp/faq.html).
- A new WEB page on academic and industrial CADP training was created. This page contains numerous pointers worldwide (Belgium, Canada, France, Israel, Japan, Romania, United Kingdom, and USA) to online materials for CADP teaching (see http://www.inrialpes.fr/vasy/cadp/training).
- A new WEB page containing references to available tutorials, slides, papers, books, and WEB sites about LOTOS and CADP was created (see http://www.inrialpes.fr/vasy/cadp/tutorial).
- Finally, most of the presentation slides produced by VASY members during the past years were made available online.

9.2. Program Committees

In 2003, the members of VASY assumed the following responsibilities:

- In collaboration with Stefania Gnesi (IEI-CNR, Pise) and Ina Schieferdecker (GMD-FOKUS, Berlin), H. Garavel was responsible for a special issue of the SCP (*Science of Computer Programming*) journal, appeared in March 2003, which gathers the best papers of FMICS'2000.
- H. Garavel was, together with John Hatcliff (Kansas State University), program chair of TACAS'2003 (*9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Varsow, Poland, April 5–13, 2003). VASY hosted the TACAS 2003 conference WEB site.
- R. Mateescu was a program committee member of VVEIS'2003 (*First International Workshop on Verification and Validation of Enterprise Information Systems*, Angers, France, April 22, 2003).
- R. Mateescu was a program committee member of FMICS'2003 (*8th International Workshop on Formal Methods for Industrial Critical Systems*, Trondheim, Norway, June 5–7, 2003).
- H. Garavel was a program committee member of PDMC'2003 (*2nd International Workshop on Parallel and Distributed Model Checking*, Boulder, Colorado, USA, July 14, 2003).
- R. Mateescu was a program committee member of SOFTMC'2003 (*2nd Workshop on Software Model Checking*, Boulder, Colorado, USA, July 14, 2003).
- F. Lang, A. Collomb, and W. Serwe co-organized (with Marieke Huisman, INRIA Sophia-Antipolis) the "MODOCOP *Workshop on Smart Card Specification, Verification, and Testing*", INRIA Rhône-Alpes, December 4, 2003.

9.3. Lectures and Invited Conferences

In 2003, we gave talks in several international conferences and workshops (see bibliography below). Additionally:

- F. Lang gave two talks entitled "NTIF *: un modèle symbolique général pour les processus séquentiels communicants avec données*" and "*Vérification compositionnelle avec* CADP", at the LIFC laboratory (Université de Franche Comté, Besançon) on February 27, 2003.
- W. Serwe gave a talk entitled "*Abstracting Call-Stacks for Interprocedural Verification of Imperative Programs*" at the Workshop on Infinite Systems and Verification of Quantitative Properties (Grenoble) on March 5–7, 2003.
- R. Mateescu gave a talk entitled "*Vérification à la volée d'applications critiques*" at the master seminar of Ecole Universitaire d'Informatique de Grenoble on March 13, 2003.
- A. Collomb, F. Lang, and W. Serwe attended the 4th MODOCOP meeting (INRIA Sophia-Antipolis) on April 17, 2003. A. Collomb gave a talk entitled "*Vérification de propriétés de l'application porte-monnaie électronique codée en NTIF à l'aide de l'outil* TREX". W. Serwe gave a talk entitled "*Abstractions for call-stacks for interprocedural analysis*".
- R. Mateescu demonstrated the EVALUATOR 3.0 tool at the "*Forum 4i*" meeting (Alpexpo, Grenoble) on May 22, 2003.
- C. Joubert attended the "*Ecole Jeunes Chercheurs en Programmation*" (Aussois, France) on May 26–June 6, 2003. He gave a talk entitled "*Approches massivement parallèles pour l'analyse de très grands espaces d'états*".
- The annual VASY seminar was held in Saint-Pierre de Chartreuse on June 10–13, 2003. The list of talks is available from http://www.inrialpes.fr/vasy/events/seminaire_2003.html.
- A. Collomb gave a talk entitled "*Vérification de propriétés de l'application porte-monnaie électronique codée en NTIF à l'aide de l'outil* TREX" at the LIFC laboratory (Université de Franche Comté, Besançon) on June 17, 2003.

- F. Lang and W. Serwe attended the 5th MODOCOP meeting (INRIA Rennes) on September 17, 2003. F. Lang gave a talk entitled "*Compositional Verification using* CADP *of the ScalAgent Deployment Protocol for Software Components*".
- R. Mateescu demonstrated the AAL model checker (see § 6.1.5) at SFM 03–SA, the 3rd International School on Formal Methods for the Design of Computer, Communication and Software Systems: Software Architecture (Bertinoro, Italy) on September 22–27, 2003.
- A. Collomb gave a talk entitled "TREX*: A Tool for Reachability Analysis of Extended Systems*" at the LSV laboratory (Ecole Normale Supérieure de Cachan) on October 9, 2003.
- H. Garavel gave an invited talk entitled "*Combiner vérification compositionnelle et évaluation de performances avec* CADP" at the STRQDS (*Systèmes Temps Réel, Qualité de Service*) meeting (Paris) on October 30, 2003.
- H. Garavel was invited at the Dagstuhl seminar Nr. 03441 on November 2–7, 2003. He gave a talk entitled "*Combining functional verification and performance evaluation using* CADP".

9.4. 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 2003:

- C. Joubert gave the course on "*Programmation fonctionnelle* (CAML)" to 2nd year students of Université Joseph Fourier, Grenoble (33 hours).
- R. Mateescu and A. Collomb gave the course on "*Temps Réel*" to the 3rd year students of ENSIMAG (21 hours).
- F. Lang gave, jointly with Flavio Oquendo, the course on "*Méthodes formelles pour l'ingénierie des logiciels : spécification et vérification de protocoles*" to the students of the computer science DEA of Université de Savoie (24 hours).
- A. Collomb gave three courses on "*Sites Webs Documentaires*", "*Produits documentaires et services internet*", and "*Algorithmique*" at IUT 2 of Université Pierre Mendès-France (122 hours).
- R. Mateescu supervised the internship of A. Catry (Ecole Polytechnique) on "*Construction d'outils de vérification au moyen de systèmes d'équations booléennes*" from March 3 to September 30, 2003.
- F. Lang and A. Collomb supervised the internship of G. Schaeffer (Supélec Metz) on "*Extension et amélioration du compilateur* NTIF" from April 3 to September 12, 2003.
- H. Garavel was a jury member of F. Tronel's PhD thesis entitled "*Application des problèmes d'accord à la tolérance aux défaillances dans les systèmes distribués asynchrones*", defended in Rennes on December 17, 2003.
- R. Mateescu was a member of the "commission de spécialistes" at Université de Savoie (section 27).

10. Bibliography

Major publications by the team in recent years

[1] 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) », North-Holland, S. T. Vuong, editor, pages 147–162, December, 1989.

[2] 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) », series Lecture Notes in Computer Science, volume 1384, Springer Verlag, B. Steffen, editor, pages 68–84, Berlin, March, 1998, http://www.inria.fr/rrrt/rr-3352.html, Full version available as Inria Research Report RR-3352.

[3] 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) », series Lecture Notes in Computer Science, volume 2391, Springer Verlag, L.-H. Eriksson, P. A. Lindsay, editors, pages 410–429, July, 2002, http://www.inria.fr/rrrt/rr-4492.html, Full version available as Inria Research Report 4492.

[4] H. Garavel, F. Lang. *SVL: a Scripting Language for Compositional Verification.* in « Proceedings of the 21st IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2001 (Cheju Island, Korea) », IFIP, Kluwer Academic Publishers, M. Kim, B. Chin, S. Kang, D. Lee, editors, pages 377–392, August, 2001, http://www.inria.fr/rrrt/rr-4223.html, Full version available as Inria Research Report RR-4223.

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

[6] 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) », series Lecture Notes in Computer Science, volume 2304, Springer Verlag, N. Horspool, editor, pages 9–13, April, 2002.

[7] H. Garavel, R. Mateescu, I. Smarandache. *Parallel State Space Construction for Model-Checking.* in « Proceedings of the 8th International SPIN Workshop on Model Checking of Software SPIN'2001 (Toronto, Canada) », series Lecture Notes in Computer Science, volume 2057, Springer Verlag, M. B. Dwyer, editor, pages 217–234, Berlin, May, 2001, http://www.inria.fr/rrrt/rr-4341.html, Full version available as Inria Research Report RR-4341.

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

[9] H. Garavel, M. Sighireanu. *Towards a Second Generation of Formal Description Techniques – Rationale*

for the Design of E-LOTOS. in « Proceedings of the 3rd International Workshop on Formal Methods for Industrial Critical Systems FMICS'98 (Amsterdam, The Netherlands) », CWI, J.-F. GROOTE, B. LUTTIK, JOS VAN. WAMEL, editors, pages 187–230, Amsterdam, May, 1998, Invited talk.

[10] H. GARAVEL, M. SIGHIREANU. *A Graphical Parallel Composition Operator for Process Algebras.* in « Proceedings of the Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, and Protocol Specification, Testing, and Verification FORTE/PSTV'99 (Beijing, China) », IFIP, Kluwer Academic Publishers, J. WU, Q. GAO, S. T. CHANSON, editors, pages 185–202, October, 1999.

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

Books and Monographs

[12] H. GARAVEL, S. GNESI, I. SCHIEFERDECKER. *Science of Computer Programming. Special issue on the 5th International Workshop on Formal Methods for Industrial Critical Systems FMICS'2000 (Berlin, Germany).* number 3, volume 46, Elsevier, March, 2003.

[13] H. GARAVEL, J. HATCLIFF, editors, *Proceedings of the 9th International Conference on Tools and Algorithms for Construction and Analysis of Systems TACAS'2003 (Warsaw, Poland).* series Lecture Notes in Computer Science, volume 2619, Springer Verlag, April, 2003.

Doctoral dissertations and "Habilitation" theses

[14] F. TRONEL. *Application des problèmes d'accord à la tolérance aux défaillances dans les systèmes distribués asynchrones.* Ph. D. Thesis, Université de Rennes I, December, 2003.

Articles in referred journals and book chapters

[15] F. LANG. *Explaining the Lazy Krivine Machine Using Explicit Substitution and Addresses.* in « Journal of Higher Order and Symbolic Computing, special issue on Krivine's machine », 2004, to appear.

[16] R. MATEESCU. *Logiques temporelles basées sur actions pour la vérification des systèmes asynchrones.* in « Technique et Science Informatiques », number 4, volume 22, 2003, pages 461–495.

[17] R. MATEESCU, M. SIGHIREANU. *Efficient On-the-Fly Model-Checking for Regular Alternation-Free Mu-Calculus.* in « Science of Computer Programming », number 3, volume 46, March, 2003, pages 255–281.

Publications in Conferences and Workshops

[18] R. ECHAHED, F. PROST, W. SERWE. *Statically Assuring Secrecy for Dynamic Concurrent Processes.* in « Proceedings of the 5th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming PPDP'2003 (Uppsala, Sweden) », 2003.

[19] H. GARAVEL. *Défense et illustration des algèbres de processus.* in « Actes de l'Ecole d'été Temps Réel

ETR 2003 (Toulouse, France) », Institut de Recherche en Informatique de Toulouse, Z. MAMMERI, editor, September, 2003.

[20] H. HERMANNS, C. JOUBERT. *A Set of Performance and Dependability Analysis Components for CADP.* in « Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2003 (Warsaw, Poland) », series Lecture Notes in Computer Science, volume 2619, Springer Verlag, H. GARAVEL, J. HATCLIFF, editors, pages 425–430, April, 2003.

[21] C. JOUBERT. *Distributed Model Checking: From Abstract Algorithms to Concrete Implementations.* in « Proceedings of the 2nd International Workshop on Parallel and Distributed Model Checking PDMC'2003 (Boulder, Colorado, USA) », series Electronic Notes in Theoretical Computer Science, volume 89, Elsevier, L. BRIM, O. GRUMBERG, editors, 2003.

[22] 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) », series Lecture Notes in Computer Science, volume 2619, Springer Verlag, H. GARAVEL, J. HATCLIFF, editors, pages 81–96, April, 2003, http://www.inria.fr/rrrt/rr-4711.html, Full version available as INRIA Research Report RR-4711.

[23] R. MATEESCU. *On-the-Fly Verification Using CADP.* in « Proceedings of the 8th International Workshop on Formal Methods for Industrial Critical Systems FMICS'2003 (Trondheim, Norway) », series Electronic Notes in Theoretical Computer Science, volume 80, Elsevier, T. ARTS, W. FOKKINK, editors, June, 2003.

[24] 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) », series Lecture Notes in Computer Science, volume 2725, Springer Verlag, J. WARREN A. HUNT, F. SOMENZI, editors, pages 446–459, July, 2003, http://www.inria.fr/rrrt/rr-4918.html, Full version available as INRIA Research Report RR-4918.

[25] F. TRONEL, F. LANG, H. GARAVEL. *Compositional Verification Using CADP of the ScalAgent Deployment Protocol for Software Components.* in « Proceedings of the 6th IFIP International Conference on Formal Methods for Open Object-based Distributed Systems FMOODS'2003 (Paris, France) », series Lecture Notes in Computer Science, Springer Verlag, U. NESTMANN, P. STEVENS, editors, November, 2003.

Internal Reports

[26] I. ALLOUI, H. GARAVEL, R. MATEESCU, F. OQUENDO. *The ArchWare Architecture Analysis Language.* Project Deliverable, number D3.1, European project IST 2001-32360 "ArchWare", January, 2003.

[27] A. CATRY, D. CHAMPELOVIER, H. GARAVEL, R. MATEESCU. *Definition of the Architecture Analysis Formalism for Model-Checking.* Project Deliverable, number D3.3, European project IST 2001-32360 "ArchWare", June, 2003.

[28] A. CATRY, D. CHAMPELOVIER, H. GARAVEL, R. MATEESCU. *Preliminary ArchWare Architecture Analysis Tool by Model-Checking.* Project Deliverable, number D3.6a, European project IST 2001-32360 "ArchWare", December, 2003.

[29] D. Champelovier, P. Fabriani, H. Garavel. *Formal Specification of Federated Knowledge Management System (FKMS) Using a Process Algebra.* Working Document, European project IST 2001-32360 “ArchWare”, February, 2003.

[30] H. Garavel, R. Mateescu. *Enhanced Model-Checker for Architecture Analysis.* Project Deliverable, number D3.8, European project IST 2001-32360 “ArchWare”, January, 2003.

[31] B. Jeannet, W. Serwe. *Abstracting Call-Stacks for Interprocedural Verification of Imperative Programs.* Research Report, number 4904, INRIA, July, 2003, http://www.inria.fr/rrrt/rr-4904.html.

Miscellaneous

[32] G. Schaeffer. *Extension et amélioration du compilateur NTIF.* mémoire d’ingénieur en 3e année, Supélec, Metz, September, 2003.

Bibliography in notes

[33] Y. Afek, G. Brown, M. Meritt. *Lazy Caching.* in « ACM Transactions on Programming Languages and Systems », number 1, volume 15, 1993.

[34] T. Arts, J. J. S. Penas. *VoDkaV Tool: Model Checking for Extracting Global Scheduler Properties from Local Restrictions.* in « Proceedings of the 3rd International Conference on Application of Concurrency to System Design ACSD’03 (Guimaraes, Portugal) », IEEE Computer Society Press, J. Lilius, F. Balarin, editors, pages 247–248, June, 2003.

[35] G. Badoil. *Vérification de circuits : problèmes et solutions – Exemple de TestBuilder.* Mémoire de probatoire en informatique, CNAM, Grenoble, March, 2002.

[36] M. Ben-Or. *Another advantage of free choice: Completely asynchronous agreement protocols.* in « Proceedings of the 2nd Annual ACM Symposium on the Principles of Distributed Computing PODC », 1983.

[37] S. Blom, N. Ioustinova, N. Sidorova. *Timed verification with μ-CRL.* in « Proceedings of the 5th Andrei Ershov International Conference on Perspectives of System Informatics PSI’2003 (Novosibirsk, Russia) », series Lecture Notes in Computer Science, volume 2890, Springer Verlag, M. Broy, A. V. Zamulin, editors, pages 178–192, July, 2003, Also available as CWI Research Report SEN-E0312, Amsterdam, December 2003.

[38] M. Boyer, M. Sighireanu. *Synthesis and Verification of Constraints in the PGM Protocol.* in « Proceedings of the 12th International Symposium of Formal Methods Europe FME’03 (Pisa, Italy) », series Lecture Notes in Computer Science, volume 2805, Springer Verlag, K. Araki, S. Gnesi, D. Mandrioli, editors, pages 264–281, September, 2003.

[39] 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) », IFIP, Chapman & Hall, R. Gotzhein, J. Bredereke, editors, pages 435–450, October, 1996,

http://www.inria.fr/rrrt/rr-2958.html, Full version available as INRIA Research Report RR-2958.

[40] B. CHEN, S. SADAOUI. *Simulation and Verification of a Dynamic Online Auction.* in « Proceedings of the 7th IASTED International Conference on Software Engineering and Applications SEA'2003 (Marina del Rey, CA, USA) », November, 2003.

[41] 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 », number 2, volume 8, April, 1986, pages 244–263.

[42] S. GRAF. *Characterization of a Sequentially Consistent Memory and Verification of a Cache Memory by Abstraction.* in « Distributed Computing », number 2–3, volume 12, 1999, pages 75–90.

[43] 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) », series Lecture Notes in Computer Science, volume 1893, Springer Verlag, M. NIELSEN, B. ROVAN, editors, pages 383–393, Berlin, August, 2000, Available as CWI Technical Report SEN-R0008, Amsterdam, March 2000.

[44] M. HENNESSY, R. MILNER. *Algebraic Laws for Nondeterminism and Concurrency.* in « Journal of the ACM », volume 32, 1985, pages 137–161.

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

[46] S. ORZAN, J. VAN DE POL. *Verification of Distributed Dataspace Architectures.* in « Proceedings of the 5th Andrei Ershov International Conference on Perspectives of System Informatics PSI'2003 (Novosibirsk, Russia) », series Lecture Notes in Computer Science, volume 2890, Springer Verlag, M. BROY, A. V. ZAMULIN, editors, pages 192–206, July, 2003.

[47] J. PANG, W. FOKKINK, R. HOFMAN, R. VELDEMA. *Model Checking a Cache Coherence Protocol for a Java DSM Implementation.* in « Proceedings of the 8th International Workshop on Formal Methods for Parallel Programming: Theory and Applications FMPPTA'2003 (Nice, France) », IEEE Computer Society Press, M. CHARPENTIER, B. SANDERS, editors, April, 2003.

[48] M. O. RABIN. *Randomized Byzantine Generals.* in « Proceedings of the IEEE Symposium on Foundations of Computer Science », pages 403–409, 1983.

[49] Y. RAMAKRISHNA, S. SMOLKA. *Partial-Order Reduction in the Weak Modal Mu-Calculus.* in « Proceedings of the 8th International Conference on Concurrency Theory CONCUR'97 », series Lecture Notes in Computer Science, volume 1243, Springer Verlag, A. MAZURKIEWICZ, J. WINKOWSKI, editors, pages 5–24, 1997.

[50] G. SCOLLO, S. ZECCHINI. *Architectural Unit Testing in a Robot Teleoperation Case Study.* Research Report, number RR 12/2003, University of Verona, Verona, Italy, October, 2003.

[51] P. WODEY, G. CAMARROQUE, F. BARAY, R. HERSEMEULE, J.-P. COUSIN. *LOTOS Code Generation for Model Checking of STBus Based SoC: The STBus Interconnect.* in « Proceedings of the 1st ACM and IEEE

International Conference on Formal Methods and Models for Codesign MEMOCODE'03 (Mont Saint-Michel, France) », R. K. GUPTA, S. SHUKLA, J.-P. TALPIN, editors, pages 204–213, June, 2003.

[52] S. ZENNOU, M. YGUEL, P. NIEBERT. *ELSE: A new symbolic state generator for timed automata.* in « Proceedings of the 1st International Workshop on Formal Modeling and Analysis of Timed Systems FORMATS 2003 (Marseille, France) », September, 2003.

[53] J. GALVEZ LONDONO. *Analyse du flux des données dans un système parallèle.* Masters (DEA) dissertation, Institut National Polytechnique de Grenoble, June, 1993.

[54] J. VAN DE POL, M. V. ESPADA. *Verification of JavaSpacesTM Parallel Programs.* in « Proceedings of the 3rd International Conference on Application of Concurrency to System Design ACSD'03 (Guimaraes, Portugal) », IEEE Computer Society Press, J. LILIUS, F. BALARIN, editors, pages 196–205, June, 2003.