INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team VASY

Validation of Systems

Grenoble - Rhône-Alpes

Theme : Embedded and Real Time Systems

Activity Report

2010

Table of contents

VASY *is an* INRIA *project team that is also a team of the* LIG *laboratory, a joint research unit of Centre National de Recherche Scientifique, Grenoble* INP*, and Université Joseph Fourier.*

1. Team

Research Scientists

Hubert Garavel [Senior Researcher INRIA, Team Leader]
Frédéric Lang [Researcher INRIA]
Radu Mateescu [Researcher INRIA, HdR]
Wendelin Serwe [Researcher INRIA]

Faculty Member

Gwen Salaün [Associate Professor, Grenoble INP]

External Collaborators

Holger Hermanns [Saarland University, until November 30, 2010]
Etienne Lantreibecq [STMICROELECTRONICS]

Technical Staff

Iker Bellicot
Simon Bouland
Yann Genevois
Rémi Hérilier
Alain Kaufmann [until November 30, 2010]
Christine McKinty [since September 1, 2010]
Vincent Powazny
Damien Thivolle [since December 1, 2010]

PhD Students

Nicolas Coste [STMICROELECTRONICS, CIFRE grant, until January 31, 2010]
Damien Thivolle [MESR grant, until November 30, 2010]
Meriem Zidouni [BULL, CIFRE grant, until February 28, 2010]

Administrative Assistant

Helen Pouchot

2. Overall Objectives

2.1. Overview

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 exhibits *asynchronous concurrency*, i.e., any system whose behavior can be modelled as a set of parallel processes governed by interleaving semantics.

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

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

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

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

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

2.2. Highlights

The 2010 evaluation by the French national agency for the evaluation of research (AERES) awarded VASY the highest possible rating, A+, for the work accomplished in the past four years. The report noted the high quality of publications, a good level of visibility and recognition internationally, and a real success in technology transfer.

3. Scientific Foundations

3.1. 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 this 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 this case, the natural approach to verification is *model checking*, which consists in deciding whether or not the system model satisfies 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 has been 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 all their states and transitions in memory (*exhaustive* verification).
- Larger models are represented *implicitly*, by exploring only the model states and transitions needed for the verification (*on the fly* verification).

3.2. Languages and Compilation Techniques

Our research focuses on high level languages with *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 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 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).
- We contributed to the definition of E-LOTOS (*Enhanced*-LOTOS, ISO/IEC standard 15437:2001), a deep revision of LOTOS, which tries to provide a greater expressiveness (for instance, by introducing quantitative time to describe systems with real-time constraints) together with a better user friendliness. Our contributions to E-LOTOS are available on the WEB (see http://www.inrialpes.fr/vasy/elotos).
- We are also working on an E-LOTOS variant, named LOTOS NT (LOTOS *New Technology*) [12], [1], in which we can experiment with 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 enables 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. The languages differ in that LOTOS NT combines imperative and functional features, and is also simpler than E-LOTOS in some respects (static typing, operator overloading, arrays), which should make it easier to implement. We are developing several tools for LOTOS NT: a prototype compiler named TRAIAN (see § 5.2), a translator from (a subset of) LOTOS NT to LOTOS (see § 6.2.2), and an intermediate semantic model named NTIF (*New Technology Intermediate Form*) [7].

3.3. Implementation and Experimentation

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

4. Application Domains

4.1. Application Domains

The theoretical framework we use (automata, process algebras, bisimulations, temporal logics, etc.) and the software tools we develop are general enough to fit the needs of many application domains. They are applicable to virtually any system or protocol that consists of distributed agents communicating by asynchronous messages. The list of recent case studies performed with the CADP toolbox (see in particular § 6.3) illustrates the diversity of applications:

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

5. Software

5.1. The CADP Toolbox

Participants: Iker Bellicot, Simon Bouland, Hubert Garavel [contact person], Yann Genevois, Rémi Hérilier, Alain Kaufmann, Frédéric Lang, Radu Mateescu, Christine McKinty, Wendelin Serwe, Damien Thivolle.

We maintain and enhance CADP (*Construction and Analysis of Distributed Processes* – formerly known as CÆSAR/ALDÉBARAN *Development Package*) [9], a toolbox for protocols and distributed systems engineering (see http://www.inrialpes.fr/vasy/cadp). In this toolbox, we develop and maintain the following tools:

- CÆSAR.ADT [3] 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 [11] 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 [4] is a generic software environment for developing tools that explore graphs on the fly (for instance, simulation, verification, and test generation tools). Such tools can be developed independently of any particular high level language. In this respect, OPEN/CÆSAR plays a central role in CADP by connecting language-oriented tools with model-oriented tools. OPEN/CÆSAR consists of a set of 16 code libraries with their programming interfaces, such as:
  - CAESAR_GRAPH, which provides the programming interface for graph exploration
  - CAESAR_HASH, which contains several hash functions
  - CAESAR_SOLVE, which resolves boolean equation systems on the fly
  - CAESAR_STACK, which implements stacks for depth-first search exploration
  - CAESAR_TABLE, which handles tables of states, transitions, labels, etc

A number of tools have been developed within the OPEN/CÆSAR environment, among which:
  - BISIMULATOR, which checks bisimulation equivalences and preorders
  - CUNCTATOR, which performs on-the-fly steady-state simulation of continuous-time Markov chains
  - DETERMINATOR, which eliminates stochastic nondeterminism in normal, probabilistic, or stochastic systems
  - DISTRIBUTOR, which generates the graph of reachable states using several machines
  - EVALUATOR, which evaluates regular alternation-free μ-calculus formulas
  - EXECUTOR, which performs random execution
  - EXHIBITOR, which searches for execution sequences matching a given regular expression
  - GENERATOR, which constructs the graph of reachable states
  - PROJECTOR, which computes abstractions of communicating systems
  - REDUCTOR, which constructs and minimizes the graph of reachable states modulo various equivalence relations
  - SIMULATOR, XSIMULATOR, and OCIS, which allow interactive simulation
  - TERMINATOR, which searches for deadlock states

- BCG (*Binary Coded Graphs*) is both a file format for storing very large graphs on disk (using efficient compression techniques) and a software environment for handling this format. BCG also plays a key role in CADP as many tools rely on this format for their inputs/outputs. The BCG environment consists of various libraries with their programming interfaces, and of several tools, such as:
  - BCG_DRAW, which builds a two-dimensional view of a graph
  - BCG_EDIT, which allows to modify interactively the graph layout produced by BCG_DRAW
  - BCG_GRAPH, which generates various forms of practically useful graphs
  - BCG_INFO, which displays various statistical information about a graph
  - BCG_IO, which performs conversions between BCG and many other graph formats
  - BCG_LABELS, which hides and/or renames (using regular expressions) the transition labels of a graph
  - BCG_MERGE, which gathers graph fragments obtained from distributed graph construction
  - BCG_MIN, which minimizes a graph modulo strong or branching equivalences (and can also deal with probabilistic and stochastic systems)
  - BCG_STEADY, which performs steady-state numerical analysis of (extended) continuous-time Markov chains
  - BCG_TRANSIENT, which performs transient numerical analysis of (extended) continuous-time Markov chains

  The BCG environment also contain XTL (*eXecutable Temporal Language*), which is a high level, functional language for programming exploration algorithms on BCG graphs. XTL provides primitives to handle states, transitions, labels, *successor* and *predecessor* functions, etc. For instance, one can define recursive functions on sets of states, which allow to specify in XTL evaluation and diagnostic generation fixed point algorithms for usual temporal logics (such as HML [48], CTL [38], ACTL [39], etc.).
- The connection between explicit models (such as BCG graphs) and implicit models (explored on the fly) is ensured by OPEN/CÆSAR-compliant compilers, e.g.:
  - CÆSAR.OPEN, for models expressed as LOTOS descriptions
  - LNT.OPEN, for models expressed as LOTOS NT descriptions
  - BCG_OPEN, for models represented as BCG graphs
  - EXP.OPEN, for models expressed as communicating automata
  - SEQ.OPEN, for models represented as sets of execution trace
  - FSP.OPEN, for models expressed in FSP

The CADP toolbox also includes TGV (*Test Generation based on Verification*), developed by the VERIMAG laboratory (Grenoble) and the VERTECS project team at INRIA Rennes.

The CADP tools are well-integrated and can be accessed easily using either the EUCALYPTUS graphical interface or the SVL [6] scripting language. Both EUCALYPTUS and SVL provide users with an easy and uniform access to the CADP tools by performing file format conversions automatically whenever needed and by supplying appropriate command-line options as the tools are invoked.

5.2. The TRAIAN Compiler

Participants: Hubert Garavel [contact person], Frédéric Lang.

We develop a compiler named TRAIAN for translating descriptions written in the LOTOS NT language (see § 3.2) into C programs, which will be used for simulation, rapid prototyping, verification, and testing.

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

Although this version of TRAIAN is still incomplete (it does not handle LOTOS NT processes), it already has useful applications in compiler construction [8]. The recent compilers developed by the VASY project team — including AAL, CHP2LOTOS (see § 6.2.3), EVALUATOR 4.0 (see § 6.1.7), EXP.OPEN 2.0 (see § 6.1.4), FSP2LOTOS (see § 6.2.3), LNT2LOTOS (see § 6.2.2), NTIF (see § 3.2), and SVL (see § 6.1.4) — 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 traversal of abstract trees. Some involved parts of the compiler can also be written directly in C if necessary. The combined use of SYNTAX, LOTOS NT, and TRAIAN proves to be satisfactory, in terms of both the rapidity of development and the quality of the 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 Format and Libraries

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

BCG (*Binary-Coded Graphs*) is both a file format for the representation of explicit graphs and a collection of libraries and programs dealing with this format. Version 1.0 of the BCG format was recently replaced by verison 1.1, which can exploit the capabilities of 64-bit addressing.

In 2010, we continued to enhance the BCG libraries to support explicit graphs larger than a billion states, as follows:

- A bug has been corrected in the BCG_EDIT tool, which could crash in some situations.
- A bug has been corrected in the BCG_IO tool, which translated correct BCG 1.0 files to incorrect BCG 1.1 files.
- We made several optimizations in BCG_INFO, significantly improving time performance.
- The dynamic C library automatically generated with each BCG file was made more robust to file moves, calls from other directories, and the presence of special characters in the BCG file names.
- The C code generated for graphs containing many labels has been improved. The time needed to read a BCG graph containing more than $600,000$ distinct labels has been divided by two.

In addition, we continued our research on the development of a new version 2.0 of the BCG format that will further increase the compactness of the format. We implemented a multithreaded approach for data compression and experimented with many different compression algorithms through extensive testing using the high-performance computing facilities provided by INRIA.

6.1.2. OPEN/CÆSAR and the CÆSAR_SOLVE Library

Participants: Iker Bellicot, Hubert Garavel, Yann Genevois, Radu Mateescu, Wendelin Serwe.

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

In 2010, we optimized the internal memory management of the OPEN/CÆSAR TABLE_1 to reduce memory usage, and fixed several bugs in related functions. We also improved the error messages issued by the OPEN/CÆSAR shell scripts if a file is missing.

CÆSAR_SOLVE is a generic software library based on OPEN/CÆSAR 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, the minimization tool REDUCTOR and the model checkers EVALUATOR 3.5 and 4.0. The resolution method is based on boolean graphs, which provide an intuitive representation of dependencies between boolean variables, and which are handled implicitly, in a way similar to the OPEN/CÆSAR interface [4].

In 2010, we corrected a bug in the CÆSAR_SOLVE library.

The BES_SOLVE tool ($4,000$ lines of C code) enables comparison and crosschecking of the various resolution algorithms provided by the CÆSAR_SOLVE library, as well as a prototype distributed resolution algorithm. The tool constructs a boolean equation system in memory, either by reading it from a (possibly compressed) text file, or by generating it randomly according to a random parameter configuration file. Then, a boolean variable defined in some equation block of the boolean system can be solved by invoking a resolution algorithm.

In 2010, we tested BES_SOLVE intensively on several thousands of boolean equation systems. We corrected several bugs in the prototype distributed resolution algorithm, and we enhanced BES_SOLVE with the possibility of solving a list of boolean variables instead of a single one, and of generating diagnostic files for each of these variables. This enables testing of the repeated execution of boolean resolution algorithms on various variables of the same boolean equation system.

6.1.3. The EVALUATOR Tool

Participants: Hubert Garavel, Yann Genevois, Alain Kaufmann, Radu Mateescu.

EVALUATOR is a model checker that evaluates a temporal logic property on a graph represented implicitly using the OPEN/CÆSAR environment. In version 3.5 of EVALUATOR, properties are described in regular alternation-free μ-calculus, a logic built from boolean operators, possibility and necessity modalities containing regular expressions denoting transition sequences, and fixed point operators without mutual recursion between least and greatest fixed points. The input language of the tool also enables the user to define parameterized temporal operators and to group them into separate libraries. EVALUATOR works on the fly, meaning that only those parts of the implicit graph relevant to verification are explored. The model checking problem is reformulated in terms of solving a boolean equation system. A useful feature of EVALUATOR is the generation of diagnostics (examples and counterexamples) explaining why a formula is true or false.

In 2010, we continued the development of the EVALUATOR 4.0 prototype tool ($4,800$ lines of SYNTAX code, $38,700$ lines of LOTOS NT code, and $9,900$ lines of C code), which accepts as input specifications written in MCL (*Model Checking Language*), an extension of the regular alternation-free μ-calculus of EVALUATOR 3.5 with data-handling and fairness operators.

We improved the rigour of the testing for EVALUATOR 3.5 and 4.0, creating a test base of 10,000 BCG graphs and 3,800 MCL formulas. The combinations of MCL formula and BCG graph tested are no longer static: instead, every MCL formula is tested on 10 BCG graphs that are selected as the most suitable because they contain labels that correspond to action predicates in the MCL formula. This enhanced testing revealed several errors, which have been corrected.

In addition to these bug fixes, we added several enhancements to EVALUATOR 4.0:

- We improved the handling of weak (possibility and necessity) modalities containing regular formulas, which characterize sequences of arbitrary length and whose visible actions can be interspersed with subsequences of zero or more invisible transitions. These modalities generalize their counterparts proposed by Stirling in the observational modal μ-calculus, which contained only action formulas and thus were able to characterize only sequences containing a single visible transition, possibly preceded and followed by zero or more invisible transitions. The translation of weak modalities into fixed point equations was enhanced in order to reduce the number of equations, which increases in the same proportion the efficiency of verification.
- The compilation of numerical expressions (involving the natural, integer, and real data types) was enhanced by implementing implicit type conversions in order to simplify the MCL programs by reducing the number of places where the "**of**" operator was necessary to eliminate typing ambiguities.
- The back-end of EVALUATOR 3.5 was incorporated into EVALUATOR 4.0 and is invoked for evaluating temporal formulas specified using the dataless fragment of MCL.
- The manual page of EVALUATOR 4.0 (49 pages) was completed.

MCL and EVALUATOR 4.0 were used successfully for analyzing mutual exclusion protocols [26] (see § 6.3.1), communication protocols (see § 6.3.5), and π-calculus specifications [25] (see § 6.2.3).

6.1.4. *Compositional Verification Tools*

Participants: Frédéric Lang, Radu Mateescu.

The CADP toolbox contains various tools dedicated to compositional verification, among which EXP.OPEN 2.0, PROJECTOR 3.0, BCG_MIN, and SVL play a central role. EXP.OPEN 2.0 explores on the fly the graph corresponding to a network of communicating automata (represented as a set of BCG files). PROJECTOR 3.0 implements behaviour abstraction [46], [54] by taking into account interface constraints. BCG_MIN minimizes behaviour graphs modulo strong or branching bisimulation and their stochastic extensions. SVL (*Script Verification Language*) is both a high level language for expressing complex verification scenarios and a compiler dedicated to this language.

In 2010, we corrected a few bugs in some of these tools and we enhanced them as follows:

- Continuing our effort to connect high-level languages to CADP (§ 6.2), we extended the SVL language and compiler to support LOTOS NT and FSP specifications, which can now be used within SVL scripts in the same way as LOTOS specifications.
- In collaboration with Pepijn Crouzen (Saarland University, Germany), we have developed a new heuristic, named *smart reduction*, to generate the BCG graph of an EXP.OPEN network compositionally, by automatically choosing an appropriate ordering of graph compositions. This heuristic iteratively selects a subset of the BCG graphs of the network, composes them, hides as many labels as possible, and minimizes the resulting composition, until all graphs of the network have been composed. The choice of the BCG graphs to be composed at each step relies on metrics that approximate (1) the proportion of transitions of the composition that can be hidden and (2) the degree of interleaving of the composition.

We implemented smart reduction in CADP. The choice of BCG graphs to be composed at each step is computed by EXP.OPEN and a new operator named "**smart reduction**" has been added to SVL. We evaluated smart reduction on a set of networks. These experiments show that smart reduction is often more efficient than the *leaf reduction* and *node reduction* heuristics already available in SVL.

- We continued to implement the BCG_MIN 2.0 tool, a new version of BCG_MIN that uses a partition refinement algorithm based on state signatures [35].

  We corrected 3 bugs in BCG_MIN and we identified a source of inefficiency in the case of stochastic and probabilistic bisimulations, due to frequent calls to an OPEN/CÆSAR function that was usually called only once in other contexts. Improving this function reduced execution time by a factor ranging from 2 to 11 depending on the graph size and equivalence relation.

  We then tested and evaluated the performance of BCG_MIN 2.0 on more than 8,000 BCG graphs. A performance comparison between BCG_MIN 1.0 and BCG_MIN 2.0 gave us the following figures: For strong and branching bisimulations, BCG_MIN 2.0 runs 20 times faster and uses 1.3 times less memory than BCG_MIN 1.0. For the stochastic and probabilistic variants of strong and branching bisimulations, BCG_MIN 2.0 runs 578 times faster and uses 4 times less memory than BCG_MIN 1.0; for particular BCG graphs, BCG_MIN 2.0 runs up to 5,800 times faster than BCG_MIN 1.0.

  We used BCG_MIN 2.0 to reduce BCG graphs that could not be reduced in reasonable time using BCG_MIN 1.0. A BCG graph ($698,000$ states, 3.5 million transitions) provided by STMICROELECTRONICS was reduced for probabilistic branching bisimulation in less than 20 minutes, using 71 MB memory. Another BCG graph (459 million states, 3 billion transitions) provided by CEA/LETI was reduced for branching bisimulation in 3 hours, using 60 GB memory. For the largest BCG graph minimized so far (841 million states, 3.5 billion transitions), strong bisimulation reduction took less than 8 hours and 83 GB memory and branching bisimulation reduction took less than 8 hours and 132 GB memory.

  BCG_MIN 2.0 became part of CADP in March 2010, replacing BCG_MIN 1.0.

6.1.5. Performance Evaluation Tools

Participants: Hubert Garavel, Frédéric Lang, Radu Mateescu.

In addition to its verification capabilities, the CADP toolbox contains several tools dedicated to performance evaluation, namely BCG_MIN, BCG_STEADY, BCG_TRANSIENT, CUNCTATOR and DETERMINATOR. In contrast to most CADP tools that operate on labeled transition systems, these tools operate on probabilistic/stochastic models derived from discrete-time and continuous-time Markov chains.

In 2010, we corrected a bug in the BCG_STEADY and BCG_TRANSIENT tools, so that the order of columns in output is preserved. This means that in a series of experiments that invoke BCG_STEADY and BCG_TRANSIENT, the order of throughputs now remains the same.

We continued working on the CUNCTATOR tool added to CADP in 2009 ($1,700$ lines of C code), which performs on-the-fly steady-state simulation of continuous-time Markov chains. In addition to correcting a bug that occasionally caused erroneous display of the results on 64-bit systems, we enhanced CUNCTATOR with two new options enabling the user to choose between four different random number generators (instead of only one generator available previously) and to specify that a given number of states are stored in an internal cache in order to speed up simulation by avoiding the recomputation of certain sequences of internal transitions.

6.1.6. Parallel and Distributed Verification Tools

Participants: Hubert Garavel, Rémi Hérilier, Radu Mateescu.

DISTRIBUTOR performs exhaustive reachability analysis and generates the labelled transition system corresponding to a BCG graph, LOTOS program, composition expression, FSP program, LOTOS NT program, or sequence file. Additionally, this program can generate a reduced labelled transition system by applying tau-compression or tau-confluence reductions on the fly. Compared to GENERATOR and REDUCTOR, which are sequential programs executing on a single machine, DISTRIBUTOR implements a distributed algorithm (derived from [10]) that runs on several machines in a grid configuration. Each machine is used to generate and store a part of the labelled transition system. This allows DISTRIBUTOR to exploit the computing resources (memory and processors) provided by many machines.

In 2010:

- We ported DISTRIBUTOR, BCG_MERGE, and the CAESAR_NETWORK library used by distributed CADP tools to the 64-bit platforms that support CADP.
- We fixed a small number of bugs in these tools, as well as in the CAESAR_NETWORK network communication library on which they are built. We deactivated label parsing by the DISTRIBUTOR (not visible to end users), thereby improving processing time. We added options to enable or disable label parsing when using BCG_MERGE. We also fixed a bug in the CPU usage displayed by the real-time monitor of DISTRIBUTOR.

6.1.7. Other Tool Developments

Participants: Hubert Garavel, Yann Genevois, Rémi Hérilier, Frédéric Lang, Radu Mateescu, Wendelin Serwe, Damien Thivolle.

A key objective for the future of CADP is the ability to support recent computing platforms. This is a heavy task because of the number of tools in CADP, their intrinsic complexity, and their reliance upon third-party software. In 2010, we continued our efforts in this direction:

- We added support for Solaris Intel 32- and 64-bit systems, and for Mac OS X 10.6 ("Snow Leopard") systems with 64-bit kernels.
- We made changes to handle the latest versions of CYGWIN on WINDOWS and recent versions of the supported C compilers.
- We updated the installation framework and enhanced the installation documentation, including information on using the new supported platforms.
- We made minor modifications to the EUCALYPTUS user interface and error messages. We fixed a bug in the information displayed on certain recent LINUX systems. We modified the EUCALYPTUS tool, adding support for FSP and LOTOS NT specifications. We enhanced support for PDF, POSTSCRIPT, zipped files, and files of unknown format.
- We enhanced support for the EMACS, XEMACS, and JEDIT text editors in CADP, notably adding support for editing LOTOS NT files.

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

- We continued building a comprehensive validation framework, based on non-regression testing and semantical checking for the CADP tools. This framework allows functional testing of individual tools as well as integration testing for several CADP tools used together to perform complex verification scenarios on various computing platforms and using various compilers.
- We developed a number of complex, internal tools, both to improve routine testing and to facilitate support of CADP users.
- We continued gathering large collections of benchmarks (BCG graphs, boolean equation systems, μ-calculus formulas, etc.) for testing the CADP tools extensively. We defined a set of rules for managing these benchmarks, making it easier to add new test patterns automatically, and scripts to check test integrity.
- To facilitate contributions from users to our test suite, we improved the CONTRIBUTOR tool, optimizing the code to improve performance, and enhancing the user interface.
- We implemented a comprehensive suite of scheduled automatic tests that run on the PIPOL platform of INRIA. These tests, which check the 40 CADP demos, validate changes on all the supported architectures, and monitor performance changes.

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 LTSMIN toolset for manipulating LTSs [36], developed at the University of Twente (The Netherlands)
- the TEPAWSN tool environment for the design of power-aware wireless sensor networks [56], [55] developed at Xi'an Jiaotong-Liverpool University (China), Vytautas Magnus University (Lithuania), and Solari (Hong Kong)
- the ALVIS modelling language for design and verification of embedded systems [61], developed at the AGH University of Science and Technology (Krakow, Poland)
- the JTORX tool for model-based testing of software [34], developed at the University of Twente (The Netherlands)
- the ARGOS tool for analysing UML descriptions [53], developed at Graz University of Technology (Austria)
- the HENSHIN language and toolset for transformation of ECLIPSE models [32], developed at CWI Amsterdam (The Netherlands) and the Technical Universities of Marburg and Berlin (Germany)
- the TTOOL environment for system-level design space exploration [52], developed at Telecom ParisTech (Sophia Antipolis, France)
- the CLOVE and MINT tools for creating composed web and grid services [63], developed at the University of Stirling (Scotland, United Kingdom)
- the KMELIA tools for component-based systems [31], developed at the University of Nantes (France)

6.2. Languages and Compilation Techniques

6.2.1. Compilation of LOTOS

Participants: Hubert Garavel, Wendelin Serwe.

The CADP toolbox contains several tools dedicated to the LOTOS language, namely the CÆSAR.ADT compiler [3] for the data type part of LOTOS, the CÆSAR compiler [11] for the process part of LOTOS, and the CÆSAR.INDENT pretty-printer.

In 2010, in addition to fixing seven bugs in the CÆSAR and CÆSAR.ADT compilers, we improved the LOTOS dedicated tools of CADP as follows:

- We added a means to enable a user to indicate whether values of a given sort are "canonical" and will be stored automatically in a hash table the maximal size of which is specified by the user. We added further checks to detect uncanonical sorts in CÆSAR.ADT, with informative messages when such sorts are detected, enabling the user to modify these sorts to make them canonical so they can be stored in hash tables.
- We modified the string management library so that strings can now be stored in hash tables. This means it is now possible to model check programs that manipulate variable-length strings. We also fixed a bug in this library.
- We added a new option to CÆSAR and CÆSAR.ADT, which can be used to print list types using the usual user-friendly notation.
- The standard definition of the LOTOS language considers "`i`" or "`I`" as a reserved keyword denoting the internal gate and forbids its use elsewhere. The error messages issued by CÆSAR and CÆSAR.ADT when "`i`" is used incorrectly have been improved. We relaxed the rules for using "`i`" as a type, sort, operation, process, variable, or specification identifier. A new option can be used to instruct CÆSAR and CÆSAR.ADT to apply the LOTOS standard strictly, rather than the new relaxed rules.
- To facilitate evolution, we merged the code bases for CÆSAR 6.3 and CÆSAR 7.1, which had diverged over time.

6.2.2. Compilation of LOTOS NT

Participants: Hubert Garavel, Rémi Hérilier, Frédéric Lang, Christine McKinty, Vincent Powazny, Wendelin Serwe.

Regarding the LOTOS NT language — a variant of E-LOTOS created by the VASY project team — we worked along two directions:

- We continued enhancing the TRAIAN compiler (see § 5.2), which generates C code from LOTOS NT data type and function definitions. TRAIAN is distributed on the Internet (see § 9.1) and used intensively within the VASY project team as a development tool for compiler construction [8].

  In 2010, TRAIAN was essentially in maintenance mode. We did, however, correct one bug and we ported the related shell scripts to add support for the new Solaris Intel 32- and 64-bit architectures.

- The LNT2LOTOS and LPP tools convert LOTOS NT code to LOTOS, thus allowing the use of CADP to verify LOTOS NT descriptions. This tool suite has been used successfully for many different systems (see § 6.3.1, S 6.3.2, S 6.3.4, S 6.3.5, S 6.3.6, and S 6.3.7).

  In 2010, the LNT2LOTOS tool was also enhanced significantly, leading to a new stable version, 5.1, which was incorporated into CADP in January 2010. The enhancements include:

  – Several changes have been made to the treatment of infix functions and constructors.

  – A new option has been added to LNT2LOTOS and LNT.OPEN to store all the character strings in a hash table.

  – The "LNT_V1" library was enriched with new functions for manipulating strings and with support for real numbers in floating point notation.

  – Important enhancements have been made to enable LNT2LOTOS to handle external C code that is provided by the user.

  – Support for lists, sets, and ordered lists has been improved.

  – Changes were made in the syntax of LOTOS NT patterns and value expressions, avoiding syntactic ambiguities, and fixing a bug regarding unary operators used without parentheses.

  – Support for multi-module compilation was added.

  – Several improvements were made to the generated LOTOS code, and the error messages and warnings issued in the case of a serious syntax error in the source LOTOS NT code were improved. The test suite for LNT2LOTOS was improved, and many new tests were added.

  – Major improvements were made to the LNT2LOTOS Reference Manual, providing new information as well as clarifications and improvements to the existing information.

  We also developed LNT.OPEN, a shell script that automates the conversion of LOTOS NT programs to LOTOS code and provides a connection between LNT2LOTOS and the OPEN/CÆSAR environment. LNT.OPEN takes as input the principal module of a LOTOS NT specification and an OPEN/CÆSAR application program. LNT.OPEN first translates the complete LOTOS NT specification (i.e., the principal module and all included modules) into LOTOS by calling LPP and LNT2LOTOS, then compiles the generated LOTOS specification by calling CÆSAR.ADT and CÆSAR, and finally invokes the OPEN/CÆSAR application program. LNT.OPEN is now the recommended tool for using LOTOS NT specifications in conjunction with CADP.

6.2.3. Source-Level Translations between Concurrent Languages

Participants: Simon Bouland, Hubert Garavel, Rémi Hérilier, Frédéric Lang, Radu Mateescu, Gwen Salaün, Wendelin Serwe, Damien Thivolle.

Although process algebras are, from a technical point of view, the best formalism to describe concurrent systems, they are not used as widely as they could be [2]. Besides the steep learning curve of process algebras, which is traditionally mentioned as the main reason for this situation, it seems also that the process algebra community scattered its efforts by developing too many languages, similar in concept but incompatible in practice. Even the advent of two international standards, such as LOTOS (in 1989) and E-LOTOS (in 2001), did not remedy this fragmentation. To address this problem, we started investigating source-level translators from various process algebras into LOTOS or LOTOS NT, so as to widen the applicability of the CADP tools.

In 2010, in addition to the LNT.OPEN tool suite (see § 6.2.2), we worked on the following translators:

- We continued to enhance FSP.OPEN, which provides a transparent interface between FSP and the OPEN/CÆSAR environment. This tool first invokes FSP2LOTOS and then EXP.OPEN 2.0 on the generated network of LTSs. We added support for FSP files to EUCALYPTUS. We fixed a bug in FSP.OPEN that caused premature deletion of certain files when invoking OCIS or XSIMULATOR. We fixed a bug in FSP2LOTOS that caused a syntax error if the first character of the FSP input filename was not a letter. We added two options to FSP.OPEN to control the types of messages that are displayed. We also collected new examples of FSP code to enhance our test suite, organized into a package now containing 775 examples. An article on FSP2LOTOS was published in an international journal [20].
- We continued our work on the FLAC tool, which translates a FIACRE program into a LOTOS program automatically, for verification using CADP. In 2010, 5 bugs reported by users of FLAC were corrected. Those corrections led to revisions 66 to 70 of the FLAC code, which is available on the development forge dedicated to FIACRE compilers[1]. We collected new examples of FIACRE code to enhance our test suite, organized into a package now containing 73 examples. We also wrote an 8-page document explaining how each FIACRE construct (type, statement, process, component, etc.) can be translated into the LOTOS NT language.
- Our study of avionics protocols (see § 6.3.5) has shown the feasibility of a systematic translation from SDL to LOTOS NT. To help users (e.g., AIRBUS) translating their own SDL specifications into LOTOS NT, we have written a document describing translation rules that can be applied systematically. This document is structured as a set of files, each of which concerns a particular construct of SDL (variables, literals, signals, procedures, etc.). Each file consists of two sections: the first section recalls the informal semantics of the SDL construct; the second section explains how this SDL construct can be expressed in LOTOS NT.
- BPEL (*Business Process Execution Language*) [51] is a language inspired by π-calculus [57] and standardized by the OASIS consortium (led by IBM and Microsoft) to describe the orchestrations of Web services. BPEL depends on other W3C standard XML-related languages: XML Schema for data types, XPATH for data expressions, and WSDL for declaring the interfaces (communications links and link functions) of a Web service.

  Following interest expressed by research teams at MIT and the Polytechnic University of Bucharest, we designed translation rules from BPEL to LOTOS NT in order to formally verify BPEL services with CADP. We began to develop an automated translator.

  In 2010, we drastically improved our translation. We formally defined the translation rules we had sketched last year. In the process of doing so, we found inaccuracies that we corrected. A better understanding of BPEL, XML Schema, XPATH, and WSDL enabled us to include many constructions that we had originally chosen to ignore. We also improved the translation from XML Schema to LOTOS NT.

  We improved several aspects of the translator, including the addition of a command-line interface.
- We considered the π-calculus [57], a process algebra based on mobile communication. We proposed a general method for translating the finite control fragment of the π-calculus (obtained by forbidding

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

recursive invocations of an agent through parallel composition operators) into LOTOS NT. The mobile communication is encoded using the data types of LOTOS NT, each channel name being represented as a value of an enumerated data type. The binary synchronization of π-calculus is enforced by associating a LOTOS NT gate to each parallel composition operator present in the π-calculus specification and by tagging each synchronization with the unique identifiers of the sender and receiver agents. The translation preserves the operational semantics by mapping each transition of a π-calculus agent to a single transition of the resulting LOTOS NT term.

The translation was implemented in the PIC2LNT tool (900 lines of SYNTAX code, $2,300$ lines of LOTOS NT code, and 500 lines of C code), developed using the SYNTAX/TRAIAN technology. The tool was tested on 160 examples of π-calculus specifications, including most of the examples provided in the Mobility Workbench distribution. This work led to a publication in an international conference [25].

6.3. Case Studies and Practical Applications

6.3.1. Mutual Exclusion Protocols

Participants: Radu Mateescu, Wendelin Serwe.

Mutual exclusion protocols are an essential building block of concurrent systems to ensure proper use of shared resources in the presence of concurrent accesses. Many variants of mutual exclusion protocols exist for shared memory, such as Peterson's or Dekker's well-known protocols. Although the functional correctness of these protocols has been studied extensively, relatively little attention has been paid to their performance aspects.

In 2010, we considered a set of 23 mutual exclusion protocols for two processes with a shared memory. We specified each protocol in LOTOS NT, using a set of generic modules to describe shared variables and the overall architecture (in total, $2,700$ lines of LOTOS NT code). Then, we compositionally added Markov delays modelling the latencies of read/write accesses on shared variables, so as to obtain the Interactive Markov Chain (IMC) corresponding to each protocol (up to $30,000$ states and $40,000$ transitions).

We verified functional properties using the same set of MCL [15] formulas for each protocol (in total, 300 lines of MCL). The mutual exclusion property was easy to express as an MCL formula, but other properties (livelock and starvation freedom, independent progress, and unbounded overtaking) turned out to be quite involved because they belong to the μ-calculus fragment of alternation depth two; fortunately, we succeeded in expressing them using the infinite looping operator of MCL, which can be checked in linear time. Finally, using the performance evaluation tools of CADP, we minimized the IMCs modulo stochastic branching bisimulation and computed the steady-state throughputs of the critical section accesses, by varying several parameters (relative speeds of processes, ratio between the time spent in critical and non-critical sections, etc.).

These experiments enabled us to compare the protocols according to their efficiency (steady-state throughputs). We observed that symmetric protocols are more robust when the difference in execution speed between processes is large, which confirms the importance of the symmetry requirement originally formulated by Dijkstra [40]. The quantitative results corroborated those of functional verification: the presence of (asymmetric) starvation of processes, detected using temporal formulas, was clearly reflected in their steady-state throughputs. This work led to a publication in an international conference [26].

6.3.2. The FAME2 Architecture

Participants: Radu Mateescu, Meriem Zidouni.

In the context of the MULTIVAL (see § 7.2) contract, together with BULL we studied the MPI software layer and MPI benchmark applications to be run on FAME2 (*Flexible Architecture for Multiple Environments*), a CC-NUMA multiprocessor architecture developed at BULL for teraflop mainframes and petaflop computing.

In 2010, we pursued the study of the "*barrier*" primitive of MPI, which allows several parallel processes to synchronize (each process arriving at the barrier waits for all the others to arrive) before continuing their execution. Our goal was to estimate the latency of the barrier primitive (i.e., the average time taken by a process to traverse the barrier) on different topologies, different software implementations of the MPI primitives, and different cache coherency protocols. Based on the LOTOS NT specifications extended with Markov delays that we previously developed for several protocols implementing the barrier primitive, we were able to compute, using the CUNCTATOR on-the-fly steady-state simulator, the latency for the *centralized*, *combining*, and *tournament* protocols for three different topologies and configurations containing up to 16 processes.

These results were presented in the PhD thesis of Meriem Zidouni [19], defended on May 25, 2010.

6.3.3. The xSTream Architecture

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

In the context of the MULTIVAL contract (see § 7.2) together with STMICROELECTRONICS, we studied XSTREAM, a multiprocessor dataflow architecture for high-performance embedded multimedia streaming applications. In this architecture, computation nodes (e.g., filters) communicate using XSTREAM queues connected by a NOC (*Network on Chip*). An XSTREAM queue generalizes a bounded FIFO queue in two ways: it provides additional primitives (such as *peek* to consult items in the middle of the queue, which is not possible with the standard *push*/*pop* primitives of FIFO queues), and a *backlog* (extra memory) to allow the increase of the queue size when the queue overflows.

In 2010, we continued performance evaluation studies to predict latency and throughput of communication between XSTREAM queues, which are composed of two queues (called push and pop, respectively) communicating over the NOC taking advantage of the flow-control mechanism offered by XSTREAM. We consolidated our IPC (*Interactive Probabilistic Chains*) approach undertaken in 2008. This approach is inspired by Interactive Markov Chains [50], but uses probabilities instead of stochastic distributions and a central clock governing all delays.

Firstly, we used the IPC approach to compute the average throughput of the put operation for different sizes of the push and pop queues. Our experiments show that the relative size of the push and pop queues forming an XSTREAM queue might influence the average throughput of the push operation, although all XSTREAM queues with the same overall size (sum of push and pop queues) are functionally equivalent (i.e., branching bisimilar). We observed that an XSTREAM queue performs best when the sizes of both queues are as similar as possible. We also observed that larger push queues better support bursts of messages, which can be better absorbed.

Secondly, we studied the impact of the flow-control mechanism on performance. Therefore, we considered a system of two XSTREAM queues (i.e., two pairs of push and pop queues) sharing the NOC, studying the effect on the throughput of the first XSTREAM queue for various consumption rates of the other XSTREAM queue. In the case that the messages of the first XSTREAM queue are rapidly consumed, i.e., removed from the pop queue of the first XSTREAM queue, we observed that the flow-control mechanism has no impact on the throughput of the second XSTREAM queue. In the case that the messages of the first XSTREAM queue are consumed slowly, we observed that the flow-control mechanism impacts the throughput of the second XSTREAM queue. Indeed, without the flow-control mechanism, the second XSTREAM queue is slowed down proportionally to the first XSTREAM queue, whereas with the flow-control mechanism, the performance of the second XSTREAM queue even increases and tends towards the values observed for a single XSTREAM queue.

These results were presented in the PhD thesis of Nicolas Coste [18], defended on June 25, 2010.

6.3.4. The Platform 2012 Architecture

Participants: Hubert Garavel, Frédéric Lang, Etienne Lantreibecq, Wendelin Serwe.

In the context of the MULTIVAL contract (see § 7.2), STMICROELECTRONICS studied PLATFORM 2012, a many-core programmable accelerator for ultra-efficient embedded computing in nanometer technology. This flexible and configurable multi-cluster platform fabric targets a range of emerging video, imaging, and next-generation immersive multimodal applications. Configurability options include the number of clusters, the number and type of processing elements (PE) per cluster, specialization of the architecture and instruction-set of the PEs, and finally, support of hardware-accelerated PEs. The platform is supported by a rich programming environment which embodies a range of platform programming models.

In 2010, we concentrated on two blocks of PLATFORM 2012:

- HWS (*HardWare Synchronizer*) is a hardware block providing mechanisms that allow the implementation of software routines for synchronization between execution threads of several processors. One of these mechanisms is the System Messenger based on asynchronous message queues. STMICROELECTRONICS decided to analyze the correctness of the System Messenger, in particular to verify properties such as correct routing and absence of message loss.

  The System Messenger comes with a set of usage rules that must be respected by the software, otherwise there are no correctness guarantees whatsoever. Therefore, we advised STMICROELECTRONICS to use a constraint-oriented modelling style, i.e., to represent each usage constraint as a process to be composed in parallel with the System Messenger and the other constraints. We further reduced the size of the model by applying data abstractions (e.g., to reduce the number of bits per message) and symmetry arguments (e.g., to limit the number of processors). We also added processes to model the application software, considering two scenarios, namely n senders with one receiver, and one sender with n receivers. This led to a LOTOS NT model (eleven modules, 300 lines), for which we generated the state spaces (about $2,000$ states and $3,000$ transitions for each scenario). For all scenarios, we verified that the messages sent from the same sender to the same receiver are received in the same order. For the n-senders-one-receiver scenario, we also verified that no message is lost, and for the one-sender-n-receivers scenario, we also verified that the messages are correctly routed to their destination.

- DTD (*Dynamic Task Dispatcher*) is a hardware block that dispatches a set of application tasks on a set of PEs. It is called dynamic because each task itself might add tasks to the set of those to be dispatched by the DTD. The DTD is synthesized from a C++ model, optimized to generate an efficient hardware block. Due to the intrinsic complexity of the DTD, STMICROELECTRONICS was interested in the co-simulation of this C++ code with a formal model of the DTD.

  In a first step, we developed a LOTOS NT model of the DTD capable of handling four PEs ($1,200$ lines of LOTOS NT). We also modelled as LOTOS NT processes the different sets of tasks corresponding to various applications. To express the operations provided by the DTD, we had to include a call-stack in the model of each PE, as a means of circumventing the static-control constraints of CÆSAR forbidding recursion over parallel composition. STMICROELECTRONICS judged LOTOS NT to be essential in modelling the DTD, because using LOTOS instead would have been extremely difficult, requiring complex continuations with numerous parameters.

  Then, we generated the state space (about $20,000$ states and $80,000$ transitions) for a simple application (containing one fork on two PEs), and verified several properties, such as the absence of deadlocks, and the correctness of the assertions inserted in the model. This allowed us to point out a difference between our implementation and the one from the architect, highlighting a high sensibility on the order of terms in an equation, revealing an under-specified mechanism. We also verified the correctness of a complex optimization.

  Having gained confidence in the LOTOS NT model, we wrote a program to automatically generate a LOTOS NT model of the DTD capable of handling n PEs. For $n = 4$, this program yielded exactly the model discussed above. Using this program, we then generated a LOTOS NT model for $n = 16$, i.e., the number of PEs handled by the C++ model used for synthesis. We applied the EXEC/CÆSAR framework to co-simulate the C++ and LOTOS NT models of the DTD, a challenge being the connection of the asynchronous LOTOS NT model with the synchronous C++ model, because one step of the C++ model corresponds, in general, to several transitions of the LOTOS NT model.

These case studies enabled us to discover and correct several bugs in CADP.

6.3.5. The Airbus Avionics Case Studies

Participants: Simon Bouland, Hubert Garavel, Wendelin Serwe, Damien Thivolle.

In the context of the TOPCASED project (see § 7.3), we studied how CADP can be used to verify avionics protocols. We have addressed case-studies provided to us by AIRBUS:

- We continued the study [13] of a ground/plane communication protocol based on TFTP (*Trivial File Transfer Protocol*) that was started in 2008. This protocol was specified as an automaton described in SAM [64], a graphical synchronous language developed by AIRBUS. To verify this protocol using CADP, a LOTOS NT specification was produced for an entire system in which two synchronous TFTP automata execute asynchronously and communicate with each other using UDP (*User Datagram Protocol*) links. This is a typical example of a GALS (*Globally Asynchronous, Locally Synchronous*) system. The TFTP automata were produced using a systematical translation from SAM to LOTOS NT. The UDP links, which may lose, duplicate and/or reorder messages, were modelled as bounded FIFO queues and bag data structures. We specified correctness properties in temporal logic and verified them using the EVALUATOR 3.5 and 4.0 model checkers of CADP, revealing a number of errors. Simulations were carried out using the EXECUTOR tool of CADP to quantify the impact of these errors on the overall protocol performance. The LOTOS NT code was optimized, and SVL scripts were generated to automate the approach. We used the 64-bit versions of CADP to generate larger BCG graphs (up to 1.5 billion states and 6.8 billion transitions), and enhanced the model checking approach by using label hiding.

  In 2010, we continued our work on this case study, notably by rewriting certain correctness properties more concisely. An extended version of [13] was produced. This case study was presented in the 2-hour introduction to TOPCASED [2], where it is described as a typical application of formal verification to avionics systems.

- AIRBUS still has SDL specifications for its A3xx airplanes, but the OBJECTGEODE tool used for their analysis is no longer supported. In this context, we studied how SDL specifications could be translated into LOTOS NT(see § 6.2.3). We considered two SDL sample specifications, developed by AIRBUS and provided to us by CS and ENSIETA:

  – The AFN (*Air Traffic System Facilities Notification*) specification consists of $12,500$ lines of SDL code (excluding comments). It has been completely translated by hand into $4,100$ lines of LOTOS NT code (excluding comments). The state space of two of the processes of the AFN has been generated: the Logon Manager ($43,591$ states and $91,875$ transitions) and the CAD Manager ($36,710$ states and $78,337$ transitions). Also, the interactive simulation tool OCIS has been used to simulate the AFN specification, which allowed us to verify the existence of particular traces.

  – The CPDLC (*Controller Pilot Data-link Communication*) specification consists of $57,200$ lines of SDL code (comments excluded). The manual translation into LOTOS NT is still in progress, but $18,700$ lines of LOTOS NT code (comments excluded) have already been generated and compiled. To date, this is the largest LOTOS NT specification ever written.

6.3.6. The Synergy Reconfiguration Protocol

Participants: Gwen Salaün, Hubert Garavel.

[2] http://gforge.enseeiht.fr/docman/view.php/52/3627/TOPCASED-presentation-2h.pdf

A major factor in the complexity of modern component-based software systems is their ability to dynamically reconfigure themselves as directed by changing circumstances. While expressing a desired reconfiguration is relatively simple, actually evolving a running system, without shutting it down, is complex, especially when considering failures that may happen during the reconfiguration process. At the heart of this reconfiguration capability lies the *reconfiguration protocol* that is responsible for incrementally and correctly evolving a running system. This protocol is implemented in the SYNERGY virtual machine, the prototype of an ongoing research programme on reconfigurable and robust component-aware virtual machines.

In 2010, we worked with Fabienne Boyer and Olivier Gruber (SARDES project-team) who designed the SYNERGY virtual machine. We specified the reconfiguration protocol using LOTOS NT and verified it with CADP. The specification in LOTOS NT consists of three parts: data types (300 lines), functions (2500 lines), and processes (900 lines). From this protocol specification, the current configuration, and the target configuration required after reconfiguration, CADP generated LTSs describing all the possible executions of the protocol.

In a second step, we used these LTSs to verify three facets of the protocol:

- We identified 8 structural invariants, and we checked that they are preserved during reconfiguration.
- We specified reconfiguration grammars ensuring that components respect the correct ordering of actions throughout the protocol. We verified that, for each component involved in a system under reconfiguration, its grammar is never violated. This is checked using hiding and reduction techniques on the whole state space to keep only operations corresponding to that component, and then checking (using the BISIMULATOR equivalence checker) that the resulting LTS is branching equivalent to the grammar.
- Because the two checks described above might not detect subtle errors that can occur in the specification (such as forbidden sequences of actions), we used temporal properties to complement these checks by analysing the application order of operations during the protocol execution. We formulated 14 μ-calculus properties that the protocol must satisfy and verified them with EVALUATOR.

Experiments were conducted on more than 200 hand-crafted configuration examples, ranging from simple to pathological. The formal analysis of the protocol and its specification over several iterations led us to make numerous revisions and improvements to the protocol itself and to its specification as we gained a better understanding of the finer points. We made 16 successive versions of the specification in all, revising several parts of the protocol, including introduction of two additional wire/unwire phases (a single wire/unwire phase was originally defined), several corrections of the failure propagation algorithm, and several corrections in the reconfiguration grammar and structural invariants. These iterations enabled us to fix bugs in some pathological cases that would have been impossible to identify manually. A paper presenting this experience has been submitted to an international conference.

6.3.7. Queuing Networks

Participants: Hubert Garavel, Holger Hermanns, Radu Mateescu.

QNAP [65] is a software application for building, manipulating, and solving queuing network models. It contains several resolution algorithms for model description, analysis, and presentation of results. BULL has been using QNAP for a long time to carry out performance analysis, and is investigating options for replacing it.

In 2010, we undertook a comparative study of QNAP and CADP, by considering two examples of QNAP models:

- A memory allocation model [65] consisting of a set of terminals and three customer stations (one CPU and two disks). We specified several variants of this model in LOTOS NT, differing in how the synchronization between stations is done and how the customer queues are implemented. Using the CUNCTATOR steady-state simulator, we were able to compute throughput values that were very similar (identical to two decimal places) to those in [65], which shows the capability of IMCs to model basic queuing network systems.

- A larger example provided by BULL (3,300 lines of QNAP code) modelling a realistic system consisting of a multiprocessor architecture, an MPI middleware layer, and a LINPACK application running on top of them. We studied how this queuing network model could be reformulated as a set of communicating LOTOS NT processes. This also suggested ideas for extending the performance analysis tools of CADP in order to handle models with distribution laws that are not exponential.

6.3.8. Other Case Studies

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

- experimentation on learning algorithms [59]
- comparison of model checkers for ERLANG [47]
- performance evaluation for NoCs [44], [45]
- translation validation of multi-clocked SIGNAL specifications [58]
- verification of web service composition [41], [42]
- model checking self-stabilizing systems [33]
- formal description and validation of production workflows [37]
- verification of the behavioural properties for group communications [30]
- automated formal analysis of CRESS web and grid services [62]
- distributed model checking for group-based applications [49]
- automated analysis of TOOLBUS scripts [43]
- validation of semantic composability [60]

7. Contracts and Grants with Industry

7.1. The EC-MOAN Project

Participants: Hubert Garavel, Radu Mateescu.

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

EC-MOAN started on February 1st, 2007 and completed in January 2010. In 2010, our efforts focused on the implementation and experimentation of on-the-fly reductions of automata modulo weak τ-confluence relations, which can improve the performance of verification by several orders of magnitude. This work led to an article accepted for publication in an international journal.

7.2. The Multival Project

Participants: Nicolas Coste, Hubert Garavel, Rémi Hérilier, Holger Hermanns, Alain Kaufmann, Frédéric Lang, Etienne Lantreibecq, Radu Mateescu, Christine McKinty, Vincent Powazny, Wendelin Serwe, Meriem Zidouni.

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

MULTIVAL started in December 2006 and was extended until May 2011. In 2010, we focused our activities on the enhancement of CADP (see § 6.2.2 and § 6.2.3) and on case studies in collaboration with our partners to verify and predict the performance of the architectures FAME2 (see § 6.3.2), xSTREAM (see § 6.3.3), and PLATFORM 2012 (see § 6.3.4).

7.3. The Topcased Project

Participants: Simon Bouland, Hubert Garavel, Frédéric Lang, Wendelin Serwe, Damien Thivolle.

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

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

TOPCASED started in August 2006 and completed in December 2010. In 2010, we enhanced the FLAC translator from FIACRE to LOTOS (see § 6.2.3) and we worked on several case studies provided to us by CS and ENSIETA (see § 6.3.5).

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

8. Other Grants and Activities

8.1. National Collaborations

From 2004 to 2010, the VASY project team played an active role in the joint research center between INRIA Rhône-Alpes and the LETI laboratory of CEA-Grenoble. In collaboration with LETI scientists (Edith Beigné, François Bertrand, Fabien Clermidy, Virang Shah, Yvain Thonnart, and Pascal Vivet), VASY developed software tools for the design of asynchronous circuits and architectures such as GALS (*Globally Asynchronous Locally Synchronous*), NOCs (*Networks on Chip*), and SOCs (*Systems on Chip*). In 2010, this collaboration was pursued as part of the MULTIVAL project (see § 7.2).

Additionally, we collaborated in 2010 with the following INRIA project teams:

- INRIA Sophia Antipolis (Eric Madelaine)
- SARDES team (Fabienne Boyer, Olivier Gruber, and Noël de Palma)
- POP-ART team (Gregor Goessler)

8.2. European Collaborations

The VASY project team is member of the FMICS (*Formal Methods for Industrial Critical Systems*) working group of ERCIM (see http://www.inrialpes.fr/vasy/fmics). From July 1999 to July 2001, H. Garavel chaired this working group. Since July 2002, he has been a member of the FMICS Board, and is in charge of dissemination actions.

In addition to our partners in aforementioned contractual collaborations, we had scientific relations in 2010 with several European universities and research centers, including:

- Imperial College (Jeff Kramer and Jeff Magee)
- Saarland University (Jonathan Bogdoll, Pepijn Crouzen, Arnd Hartmanns, and Holger Hermanns)
- University of Malaga, Spain (Carlos Canal, Javier Cubo, Francisco Duran, Jose Antonio Martin, Meriem Ouederni, and Ernesto Pimentel)
- Polytechnic University of Bucharest (Valentin Cristea)

8.3. International Collaborations

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

In 2010, we had scientific relations with the University of California at Santa Barbara, USA (Tevfik Bultan).

8.4. Visits and Exchanges

In 2010, we had the following scientific exchanges:

- Pepijn Crouzen (Saarland University, Saarbrücken, Germany) visited us on March 15–17, 2010, and gave a talk entitled "*Architectural Dependability Evaluation with* ARCADE".
- Meriem Ouederni (University of Málaga, Spain) visited us January 4 to April 9, 2010, when she gave a presentation on her work, and October 1–31, 2010.
- Eric Madelaine and Raluca Halalai (INRIA Sophia Antipolis) visited us on September 2–3, 2010.
- Philippe Dhaussy and Amine Raji (ENSIETA) attended the VASY seminar on June 21, 2010, and gave a talk entitled "*Contribution à la mise en oeuvre de techniques de validation formelle de logiciels embarqués*".
- Valentin Cristea (Polytechnic University of Bucharest) attended the VASY seminar on June 22, 2010, and gave a talk entitled "*Service oriented dependable distributed systems*".
- Holger Hermanns (University of Saarland and INRIA) attended the VASY seminar on June 23, 2010, and gave a talk entitled "*When Markov Chains meet Probabilistic Automata*".
- Arnd Hartmanns (University of Saarland) attended the VASY seminar on June 23, 2010, and gave a talk entitled "*Modelling and Model-Checking with* MODEST".
- Jonathan Bogdoll (University of Saarland) attended the VASY seminar on June 23, 2010, and gave a talk entitled "*Discrete Event Simulation for* MODEST *in Practice*".
- Pepijn Crouzen (University of Saarland) attended the VASY seminar on June 23, 2010, and gave a talk entitled "*Aggregation Ordering for Massively Compositional Models*".

9. Dissemination

9.1. Software Dissemination and Internet Visibility

The VASY project team distributes two main software tools: the CADP toolbox (see § 5.1) and the TRAIAN compiler (see § 5.2). In 2010, the main facts are the following:

- We prepared and distributed 9 successive beta-versions (from 2008-h to 2008-k and from 2009-a to 2009-e "Zurich") of CADP.
- The number of license contracts signed for CADP increased from 411 to 429.
- We were requested to grant CADP licenses for 502 different computers in the world.
- The TRAIAN compiler was downloaded by 31 different sites.

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

In September 2007, we opened the "CADP Forum" (see http://www.inrialpes.fr/vasy/cadp/forum.html) for discussions regarding the CADP toolbox. By the end of December 2010, this forum had over 150 registered users and over 1070 messages had been exchanged. Since June 2009, there has been a Wikipedia page for CADP.

9.2. Program Committees

In 2010, the members of VASY took on the following responsibilities:

- H. Garavel was a program committee member for the *Workshop on Tool Building in Formal Methods* (held in conjunction with the 2nd International ABZ Conference), Orford, Quebec, Canada, February 22, 2010.
- F. Lang was a program committee member for NEPTUNE'2010 (*Nice Environment with a Process and Tools Using Norms and Example*), Toulouse, France, May 18–19, 2010.
- G. Salaün was chair of the program committee and editor of the proceedings for WCSI'10 (*International Workshop on Component and Service Interoperability*), Málaga, Spain, June 29, 2010.
- F. Lang was a program committee member for ECSA'2010 (*4th European Conference on Software Architecture*), Copenhagen, Denmark, August 23–26, 2010.
- G. Salaün was chair of the program committee and editor of the proceedings for FOCLASA'10 (*9th International Workshop on the Foundations of Coordination Languages and Software Architectures*), Paris, France, September 4, 2010,
- G. Salaün was chair of the program committee and editor of the proceedings for TAVWEB'10 (*Fourth International Workshop on Testing, Analysis and Verification of Web Software*), Antwerp, Belgium, September 21, 2010.
- G. Salaün was a program committee member for WASELF'10 (*Third Workshop on Automatic and SELF-adaptive Systems*), Valencia, Spain, September 7, 2010.
- H. Garavel and R. Mateescu were program committee members for FMICS'2010 (*15th International Workshop on Formal Methods for Industrial Critical Systems*), Antwerp, Belgium, September 20–21, 2010.
- R. Mateescu was a program committee member for PDMC'2010 (*9th International Workshop on Parallel and Distributed Methods in verifiCation*), Twente, The Netherlands, September 30–October 1st, 2010.
- H. Garavel was a program committee member for IFM'2010 (8th International Conference on Integrated Formal Methods), Nancy, France, October 11–14, 2010.
- G. Salaün was a program committee member for AVYTAT'10 (*First International Workshop on Adaptation in serVice EcosYsTems and ArchiTectures*), Crete, Greece, October 25–29, 2010.
- G. Salaün was a program committee member for WSFM'10 (*7th International Workshop on Web Services and Formal Methods*), Hoboken, New Jersey, USA, September 16–17, 2010.
- R. Mateescu was a program committee member for ECOWS'2010 (*8th International Conference on Web Services*), Ayia Napa, Cyprus, December 1–3, 2010.

9.3. Lectures and Invited Conferences

In 2010, we we gave talks in several international conferences and workshops (see bibliography below). Additionally:

- R. Mateescu participated in the 8th EC-MOAN meeting held in Amsterdam (The Netherlands) on January 17–18, 2010. He gave a talk entitled "*Sequential and Distributed On-the-Fly Property-Dependent Reductions of Automata*" on January 17, 2010.
- H. Garavel, F. Lang, R. Mateescu, and W. Serwe participated in the 14th Multival quarterly meeting held at INRIA Grenoble (France) on April 8, 2010. F. Lang gave a talk entitled "*Fast minimization using* BCG_MIN *2.0*". W. Serwe gave a talk entitled "*What is new in* LNT2LOTOS *and* LPP *version 5.0?*".
- R. Mateescu gave a talk entitled "*On-the-Fly Model Checking with* EVALUATOR *3*" at the University of Lisbon (Portugal) on May 17, 2010.
- F. Lang gave demonstrations of CADP at the "*Journées* INRIA*/Industrie*" held in Toulouse (France) on May 17, 2010.
- F. Lang participated in the NEPTUNE conference held in Toulouse (France) on May 18–19, 2010.
- H. Garavel and G. Salaün attended the SARDES seminar, (Allevard, France), June 9–10, 2010. G. Salaün gave a talk entitled "*Specifying and Verifying the* SYNERGY *Apply Protocol with* LOTOS NT *and* CADP". H. Garavel gave a talk entitled "*An Overview of* CADP *2009*".
- R. Mateescu participated in the 15th MULTIVAL plenary meeting held at CEA/LETI (Grenoble, France) on June 25, 2010, where he gave a talk entitled "*A Study of Shared-Memory Mutual Exclusion Protocols using* CADP".
- H. Garavel gave a talk entitled "*Modélisation et vérification de systèmes parallèles complexes*" to students from ENS Cachan visiting the LIG, on November 3, 2010.
- H. Garavel participated in the panel discussion at FOSSA (*Free Open Source Software for Academia*), Grenoble, on 8 November, 2010.
- F. Lang participated in a video-conference held on November 9, 2010, to discuss the opportunity of a project on the management of requirements along the software lifecycle. He gave a talk entitled "*Modélisation et vérification des exigences avec* CADP".
- H. Garavel, F. Lang, and V. Powazny attended the 16th MULTIVAL plenary meeting held at STMICROELECTRONICS (Grenoble, France) on December 16, 2010. H. Garavel gave a talk entitled "*Améliorations apportées à CADP en 2010*", F. Lang gave a talk entitled "*Smart Reduction*", and V. Powazny gave a talk entitled "*Améliorations apportées à LOTOS NT depuis juin 2010*".

9.4. Teaching

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

In 2010:

- H. Garavel, F. Lang, and W. Serwe gave, jointly with Pascal Raymond (CNRS, VERIMAG), a course on "*Méthodes formelles de développement*" to the computer science engineering students of CNAM (*Conservatoire National des Arts et Métiers*) Grenoble (27 hours).
- F. Lang and W. Serwe gave a course on "*Modélisation et Vérification des Systèmes Concurrents et Temps-Réel*" to the 3rd year students of ENSIMAG (18 hours).
- G. Salaün gave lectures on "*Algorithmics and Object-Oriented Programming*" to the 2nd year students and on "*Specification and Validation of Distributed Processes*" to the 3rd year students of ENSIMAG (64 hours).

- G. Salaün gave a master class on "*Specification, Verification, and Adaptation of Web Services*" in Málaga, Spain, in April 2010 (10 hours).
- R. Mateescu co-supervised the PhD thesis of Pedro Tiago Monteiro (University of Lisbon, Portugal), defended on May 17, 2010.
- R. Mateescu and G. Chehaibar (BULL) co-supervised the PhD thesis of Meriem Zidouni (University Joseph Fourier, Grenoble), defended on May 25, 2010.
- H. Garavel and W. Serwe supervised the PhD thesis of Nicolas Coste (University of Grenoble), defended on June 24, 2010.
- G. Salaün was a jury member for Javier Cubo's PhD thesis, defended at the University of Málaga, Spain, in December 2010.

9.5. Miscellaneous Activities

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

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

F. Lang is a member of the "*commission du développement technologique*", which is in charge of selecting R&D projects for INRIA Grenoble Rhône-Alpes.

F. Lang participated in a working group in charge of proposing a new distribution of offices among the research and administrative teams located in the INRIA building of Montbonnot. The task of this working group ended in September 2010, after several teams moved.

R. Mateescu is the correspondent of the "*Département des Partenariats Européens*" for INRIA Grenoble Rhône-Alpes.

H. Garavel participated in the selection committee for a MAÎTRE DE CONFÉRENCES position at Université Paul Sabatier (Toulouse).

G. Salaün was elected to the ENSIMAG Council (*Conseil de l'Ecole*).

10. Bibliography

Major publications by the team in recent years

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

[2] H. GARAVEL. *Reflections on the Future of Concurrency Theory in General and Process Calculi in Particular*, in "Proceedings of the LIX Colloquium on Emerging Trends in Concurrency Theory (Ecole Polytechnique de Paris, France), November 13–15, 2006", C. PALAMIDESSI, F. D. VALENCIA (editors), Electronic Notes in Theoretical Computer Science, Elsevier Science Publishers, April 2008, vol. 209, p. 149–164, Also available as INRIA Research Report RR-6368, http://hal.inria.fr/inria-00191141.

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

[4] H. GARAVEL. *OPEN/CÆSAR: An Open Software Architecture for Verification, Simulation, and Testing*, in "Proceedings of the First International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'98 (Lisbon, Portugal)", Berlin, B. STEFFEN (editor), Lecture Notes in Computer Science, Springer Verlag, March 1998, vol. 1384, p. 68–84, Full version available as Inria Research Report RR-3352, http://hal.inria.fr/inria-00073337.

[5] H. GARAVEL, H. HERMANNS. *On Combining Functional Verification and Performance Evaluation using CADP*, in "Proceedings of the 11th International Symposium of Formal Methods Europe FME'2002 (Copenhagen, Denmark)", L.-H. ERIKSSON, P. A. LINDSAY (editors), Lecture Notes in Computer Science, Springer Verlag, July 2002, vol. 2391, p. 410–429, Full version available as Inria Research Report 4492, http://hal.inria.fr/inria-00072096.

[6] H. GARAVEL, F. LANG. *SVL: a Scripting Language for Compositional Verification*, in "Proceedings of the 21st IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2001 (Cheju Island, Korea)", M. KIM, B. CHIN, S. KANG, D. LEE (editors), Kluwer Academic Publishers, August 2001, p. 377–392, Full version available as Inria Research Report RR-4223, http://hal.inria.fr/inria-00072396.

[7] H. GARAVEL, F. LANG. *NTIF: A General Symbolic Model for Communicating Sequential Processes with Data*, in "Proceedings of the 22nd IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2002 (Houston, Texas, USA)", D. PELED, M. VARDI (editors), Lecture Notes in Computer Science, Springer Verlag, November 2002, vol. 2529, p. 276–291, Full version available as Inria Research Report RR-4666, http://hal.inria.fr/inria-00071919.

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

[9] H. GARAVEL, F. LANG, R. MATEESCU, W. SERWE. *CADP 2006: A Toolbox for the Construction and Analysis of Distributed Processes*, in "Proceedings of the 19th International Conference on Computer Aided Verification CAV'2007 (Berlin, Germany)", W. DAMM, H. HERMANNS (editors), Lecture Notes in Computer Science, Springer Verlag, July 2007, vol. 4590, p. 158–163, http://hal.inria.fr/inria-00189021.

[10] 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)", Berlin, M. B. DWYER (editor), Lecture Notes in Computer Science, Springer Verlag, May 2001, vol. 2057, p. 217–234, Revised version available as INRIA Research Report RR-4341 (December 2001).

[11] H. GARAVEL, J. SIFAKIS. *Compilation and Verification of LOTOS Specifications*, in "Proceedings of the 10th International Symposium on Protocol Specification, Testing and Verification (Ottawa, Canada)", L. LOGRIPPO, R. L. PROBERT, H. URAL (editors), North-Holland, June 1990, p. 379–394.

[12] H. GARAVEL, M. SIGHIREANU. *Towards a Second Generation of Formal Description Techniques – Rationale for the Design of E-LOTOS*, in "Proceedings of the 3rd International Workshop on Formal Methods for Industrial Critical Systems FMICS'98 (Amsterdam, The Netherlands)", Amsterdam, J. F. GROOTE, B. LUTTIK, J. VAN WAMEL (editors), CWI, May 1998, p. 187–230, Invited talk.

[13] H. GARAVEL, D. THIVOLLE. *Verification of GALS Systems by Combining Synchronous Languages and Process Calculi*, in "Proceedings of the 16th International SPIN Workshop on Model Checking of Software

SPIN'2009 (Grenoble, France)", C. PASAREANU (editor), Lecture Notes in Computer Science, Springer Verlag, June 2009, vol. 5578, p. 241–260, http://hal.inria.fr/inria-00388819.

[14] C. HELMSTETTER, O. PONSINI. *A Comparison of Two SystemC/TLM Semantics for Formal Verification*, in "Proceedings of the 6th ACM-IEEE International Conference on Formal Methods and Models for Codesign MEMOCODE'2008 (Anaheim, CA, USA)", IEEE Computer Society Press, June 2008, p. 59–68, http://hal.inria.fr/inria-00275456.

[15] R. MATEESCU, D. THIVOLLE. *A Model Checking Language for Concurrent Value-Passing Systems*, in "Proceedings of the 15th International Symposium on Formal Methods FM'08 (Turku, Finland)", J. CUELLAR, T. MAIBAUM, K. SERE (editors), Lecture Notes in Computer Science, Springer Verlag, May 2008, n^{o} 5014, p. 148–164, http://hal.inria.fr/inria-00315312/fr/.

[16] O. PONSINI, W. SERWE. *A Schedulerless Semantics of TLM Models Written in SystemC via Translation into LOTOS*, in "Proceedings of the 15th International Symposium on Formal Methods FM'08 (Turku, Finland)", J. CUELLAR, T. MAIBAUM, K. SERE (editors), Lecture Notes in Computer Science, Springer Verlag, May 2008, n^{o} 5014, p. 278–293, http://hal.inria.fr/inria-00259944.

[17] D. THIVOLLE, H. GARAVEL, X. CLERC. *Présentation du langage SAM d'Airbus*, INRIA/VASY, 16 pages, 2008, https://gforge.enseeiht.fr/docman/view.php/33/2745/SAM.pdf.

Publications of the year

Doctoral Dissertations and Habilitation Theses

[18] N. COSTE. *Vers la prédiction de performance de modèles compositionnels dans les architectures GALS*, University of Grenoble, June 2010, http://hal.inria.fr/tel-00538425/en.

[19] M. ZIDOUNI. *Modélisation et analyse des performances de la bibliothèque MPI en tenant compte de l'architecture matérielle*, Université Joseph-Fourier - Grenoble I, May 2010, http://hal.inria.fr/tel-00526164/en.

Articles in International Peer-Reviewed Journal

[20] F. LANG, G. SALAÜN, R. HÉRILIER, J. KRAMER, J. MAGEE. *Translating FSP into LOTOS and Networks of Automata*, in "Formal Aspects of Computing", November 2010, vol. 22, p. 681-711, http://hal.inria.fr/hal-00533808/en.

International Peer-Reviewed Conference/Proceedings

[21] N. COSTE, H. GARAVEL, H. HERMANNS, F. LANG, R. MATEESCU, W. SERWE. *Ten Years of Performance Evaluation for Concurrent Systems Using CADP*, in "4th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation ISoLA 2010", Grèce Amirandes, Heraclion, T. MARGARIA, B. STEFFEN (editors), October 2010, vol. 6416, p. 128-142, http://hal.inria.fr/inria-00532914/en.

[22] J. CUBO, E. PIMENTEL, G. SALAÜN, C. CANAL. *Handling Data-Based Concurrency in Context-Aware Service Protocols*, in "FOCLASA'10", France Paris, EPTCS, 2010, vol. 30, p. 62-76, http://hal.inria.fr/inria-00539024/en.

[23] J. CÁMARA, J. A. MARTIN, G. SALAÜN, C. CANAL, E. PIMENTEL. *A Case Study in Model-based Adaptation of Web Services*, in "Proc. of ISOLA'10", Grèce Heraclion, Crete, Springer, 2010, vol. 6416, p. 112–126, http://hal.inria.fr/inria-00538968/en.

[24] J. CÁMARA, J. A. MARTIN, G. SALAÜN, C. CANAL, E. PIMENTEL. *Semi-automatic Specification of Behavioural Service Adaptation Contracts*, in "FESCA'10", Chypre Paphos, ENTCS, 2010, vol. 264(1), p. 19-34, http://hal.inria.fr/inria-00539116/en.

[25] R. MATEESCU, G. SALAÜN. *Translating Pi-Calculus into LOTOS NT*, in "Integrated Formal Methods - IFM 2010", France Nancy, D. MERY, S. MERZ (editors), Lecture Notes in Computer Science, Springer Berlin / Heidelberg, October 2010, vol. 6396, p. 229-244, http://hal.inria.fr/inria-00524586/en.

[26] R. MATEESCU, W. SERWE. *A Study of Shared-Memory Mutual Exclusion Protocols using CADP*, in "15th International Workshop on Formal Methods for Industrial Critical Systems FMICS'2010", Belgique Antwerp, September 2010, http://hal.inria.fr/inria-00532897/en.

[27] M. OUEDERNI, G. SALAÜN. *Tau Be or not Tau Be? - A Perspective on Service Compatibility and Substitutability*, in "Proc. of WCSI'10", Espagne Malaga, EPTCS, 2010, vol. 37, p. 57-69, http://hal.inria.fr/inria-00539099/en.

[28] M. OUEDERNI, G. SALAÜN, E. PIMENTEL. *Quantifying Service Compatibility: A Step Beyond the Boolean Approaches*, in "ICSOC'10", États-Unis San Francisco, Springer, 2010, To appear., http://hal.inria.fr/inria-00538963/en.

[29] G. SALAÜN. *Analysis and Verification of Service Interaction Protocols - A Brief Survey*, in "TAV-WEB'10", Belgique Antwerp, EPTCS, 2010, vol. 35, p. 75-86, http://hal.inria.fr/inria-00539017/en.

References in notes

[30] R. AMEUR-BOULIFA, L. HENRIO, E. MADELAINE. *Behavioural models for group communications*, in "Proceedings of the International Workshop on Component and Service Interoperability, WICS'10 (Malaga, Spain)", 2010.

[31] P. ANDRE, G. ARDOUREL, C. ATTIOGBÉ, A. LANOIX. *Using Assertions to Enhance the Correctness of Kmelia Components and their Assemblies*, in "6th International Workshop on Formal Aspects of Component Software (FACS 2009), Eindhoven, The Netherlands", Elsevier, October 2009, vol. 263, p. 5–30.

[32] T. ARENDT, E. BIERMANN, S. JURACK, C. KRAUSE, G. TAENTZER. *Henshin: Advanced Concepts and Tools for In-Place EMF Model Transformations*, in "Model Driven Engineering Languages and Systems, Lecture Notes in Computer Science", 2010, vol. 6394, p. 121–135.

[33] A. BASU, B. BONAKDARPOUR, M. BOZGA, J. SIFAKIS. *Systematic Correct Construction of Self-stabilizing Systems: A Case Study*, in "Stabilization, Safety, and Security of Distributed Systems; Lecture Notes in Computer Science", 2010, vol. 6366/2010, p. 4–18.

[34] A. BELINFANTE. *JTorX: A tool for on-line model-driven test derivation and execution*, in "Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010 (Paphos, Cyprus), Lecture Notes in Computer Science", Springer Verlag, March 2010, vol. 6015, p. 266–270.

[35] S. BLOM, S. ORZAN. *Distributed State Space Minimization*, in "Springer International Journal on Software Tools for Technology Transfer (STTT)", 2005, vol. 7, n^{o} 3, p. 280–291.

[36] S. BLOM, J. VAN DE POL, M. WEBER. *LTSmin: Distributed and Symbolic Reachability*, in "Computer Aided Verification; Lecture Notes in Computer Science", 2010, vol. 6174, p. 354–359.

[37] V. CARCHIOLO, A. LONGHEU, M. MALGERI. *Using Lotos in Workflow Specification*, in "Proceedings of ICEIS (3)'03", 2003, p. 364-369.

[38] 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", April 1986, vol. 8, n^{o} 2, p. 244–263.

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

[40] E. W. DIJKSTRA. *Cooperating Sequential Processes*, Technological University, Eindhoven, the Netherlands, 1965.

[41] C. DUMEZ, M. BAKHOUYA, J. GABER, M. WACK. *Formal Specification and Verification of Service Composition using LOTOS*, in "Proceedings of the 7th ACM International Conference on Pervasive Services (ICPS 2010) (Berlin, Germany)", ACM, July 2010.

[42] C. DUMEZ. *Approche dirigée par les modèles pour la spécification, la vérification formelle et la mise en oeuvre de services Web composés*, Universite de Technologie de Belfort-Montbeliard, August 2010, http://hal.archives-ouvertes.fr/tel-00515130/.

[43] W. FOKKINK, P. KLINT, B. LISSER, Y. S. USENKO. *Automated Translation and Analysis of a ToolBus Script for Auctions*, in "Fundamentals of Software Engineering", 2010, vol. 5961, p. 308–323.

[44] S. FOROUTAN, Y. THONNART, R. HERSEMEULE, A. JERRAYA. *Analytical computation of packet latency in a 2D-mesh NoC*, in "Joint IEEE North-East Workshop on Circuits and Systems and TAISA Conference, 2009. NEWCAS-TAISA '09 (Toulouse, France)", IEEE, July 2009, p. 1–4.

[45] S. FOROUTAN, Y. THONNART, R. HERSEMEULE, A. JERRAYA. *A Markov chain based method for NoC end-to-end latency evaluation*, in "IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum (IPDPSW), (Atlanta, Georgia, USA)", IEEE, April 2010, p. 1–8.

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

[47] Q. GUO, J. DERRICK, C. B. EARLE, L.-A. FREDLUND. *Model-Checking Erlang - A Comparison between EtomCRL2 and McErlang*, in "Testing - Practice and Research Techniques, 5th International Academic and Industrial COnference, TAIC PART 2010, Windor, UK", Springer Verlag, September 2010, p. 23–38.

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

[49] L. HENRIO, E. MADELAINE. *Experiments with distributed Model-Checking of group-based applications*, INRIA Sophia-Antipolis. Presented at the Sophia-Antipolis Formal Analysis Group 2010 Workshop, SAFA2010, October 2010.

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

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

[52] D. KNORRECK, L. APVRILLE, R. PACALET. *Formal system-level design space exploration*, in "10th Annual International Conference on New Technologies of Distributed Systems (NOTERE), Tozeur, Tunisia", IEEE, June 2010, p. 1–8.

[53] W. KRENN, R. SCHLICK, B. K. AICHERNIG. *Mapping UML to Labeled Transition Systems for Test-Case Generation*, in "Formal Methods for Components and Objects, Lecture Notes in Computer Science", 2010, vol. 6286, p. 186–207.

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

[55] K. MAN, T. KRILAVICIUS, T. VALLEE, H. LEUNG. *TEPAWSN: A Formal Analysis Tool for Wireless Sensor Networks*, in "International Journal of Research and Reviews in Computer Science", 2010, vol. 1, p. 24–26.

[56] K. MAN, T. VALLEE, H. LEUNG, M. MERCALDI, J. VAN DER WULP, M. DONNO, M. PASTRNAK. *TEPAWSN - A tool environment for Wireless Sensor Networks*, in "Proceedings of the 4th IEEE Conference on Industrial Electronics and Applications, ICIEA 2009. (Xi'an, China)", IEEE, May 2009, p. 730–733.

[57] R. MILNER. *Communicating and Mobile Systems: the Pi-Calculus*, Cambridge University Press, 1999.

[58] J. C. PERALTA, T. GAUTIER, L. BESNARD, P. LE GUERNIC. *LTSs for Translation Validation of (multi-clocked) SIGNAL specifications*, in "8th IEEE/ACM International Conference on Formal Methods and Models for Codesign, MEMOCODE (Grenoble, France)", IEEE, July 2010, p. 199–208.

[59] H. RAFFELT, B. STEFFEN, T. BERG, T. MARGARIA. *LearnLib: a framework for extrapolating behavioral models*, in "International Journal on Software Tools for Technology Transfer (STTT), Special Section on FMICS 05 (Lisbon, Portugal)", Springer Verlag, April 2009, p. 393–407.

[60] C. SZABO, Y. M. TEO. *On Validation of Semantic Composability in Data-Driven Simulation*, in "IEEE Workshop on Principles of Advanced and Distributed Simulation (PADS)", IEEE, May 2010, p. 1–8.

[61] M. SZPYRKA, P. MATYASIK, R. MRÓWKA. *Alvis approach to Hexor robot controller development*, in "Proceedings of the 17th International Conference on Mixed Design of Integrated Circuits and Systems (MIXDES) (Wroclaw, Poland)", IEEE, June 2010, p. 595–600.

[62] L. TAN. *Case Studies Using CRESS to Develop Web and Grid Services*, Department of Computing Science and Mathematics, University of Stirling, December 2009.

[63] L. TAN. *An Integrated Methodology for Creating Composed Web/Grid Services*, University of Stirling, Scotland, UK, September 2010.

[64] D. THIVOLLE, H. GARAVEL, X. CLERC. *Présentation du langage SAM d'Airbus*, 2008, INRIA/VASY, 16 pages.

[65] M. VERAN, D. POTIER. *QNAP 2:A portable environment for queueing systems modelling*, INRIA, 06 1984, n^o RR-0314, http://hal.inria.fr/inria-00076243/en/.