INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team VASY

Validation of Systems

Rhône-Alpes

THEME COM

Table of contents

1. Team

Head of Team
Hubert Garavel [DR2 INRIA]

Administrative Assistant
Elodie Toihein

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

Software Engineers
David Champelovier
Marie Vidal [since September 1st, 2006]

Post-Doctoral Fellows
Gwen Salaün [until December 10, 2006]
Olivier Ponsini [since October 2nd, 2006]

Ph. D. Students
Christophe Joubert [until January 4, 2006]
Jan Stoecker [since September 1st, 2006]

Student Interns
Jérôme Fereyre [CNAM Grenoble, until November 29, 2006]
Nathalie Lépy [CNAM Grenoble, until August 11, 2006]
Abdul Malik Khan [Université Joseph Fourier (Grenoble), until June 30, 2006]
Damien Thivolle [EPITA Paris, until June 30, 2006]

2. Overall Objectives

2.1. Introduction

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

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

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

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

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

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

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

2.2. Models and Verification Techniques

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

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

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

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

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

2.3. Languages and Compilation Techniques

Our research focuses on high level languages with an *executable* and *formal* semantics. The former requirement stems from enumerative verification, which relies on the efficient execution of high level descriptions. The latter requirement states that languages lacking a formal semantics are not suitable for safety critical systems (as language ambiguities usually lead to interpretation divergences between designers and implementors). Moreover, enumerative techniques are not always sufficient to establish the correctness of an infinite system (they only deal with finite abstractions); one might need theorem proving techniques, which only apply to languages with a formal semantics.

We are working on several languages with the above properties:

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

- 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*) [10], [1], in which we can experiment new ideas more freely than in the constrained framework of an international standard. Like E-LOTOS, LOTOS NT consists of three parts: a *data part*, which allows the description of data types and functions, a *process part*, which extends the LOTOS process algebra with new constructs such as exceptions and quantitative time, and *modules*, which provide for structure and genericity. Both languages differ in that LOTOS NT combines imperative and functional features, and is also simpler than E-LOTOS in some respects (static typing, operator overloading, arrays), which should make it easier to implement. We are developing several tools for LOTOS NT: a prototype compiler named TRAIAN (see § 4.2), a translator from (a subset of) LOTOS NT to LOTOS (see § 5.2.2), and an intermediate semantic model named NTIF (*New Technology Intermediate Form*) [6].

2.4. Implementation and Experimentation

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

3. Application Domains

3.1. Application Domains

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

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

4. Software

4.1. The CADP Toolbox

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

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

- CÆSAR.ADT [2] is a compiler that translates LOTOS abstract data types into C types and C functions. The translation involves pattern-matching compiling techniques and automatic recognition of usual types (integers, enumerations, tuples, etc.), which are implemented optimally.
- CÆSAR [9] is a compiler that translates LOTOS processes into either C code (for rapid prototyping and testing purposes) or finite graphs (for verification purpose). The translation is done using several intermediate steps, among which the construction of a Petri net extended with typed variables, data handling features, and atomic transitions.
- OPEN/CÆSAR [3] is a generic software environment for developing tools that explore graphs on the fly (for instance, simulation, verification, and test generation tools). Such tools can be developed independently from any particular high level language. In this respect, OPEN/CÆSAR plays a central role in CADP by connecting language-oriented tools with model-oriented tools. OPEN/CÆSAR consists of a set of 16 code libraries with their programming interfaces, such as:
  - CAESAR_GRAPH, which provides the programming interface for graph exploration,
  - CAESAR_HASH, which contains several hash functions,
  - CAESAR_SOLVE, which resolves boolean equation systems on the fly,
  - CAESAR_STACK, which implements stacks for depth-first search exploration,
  - CAESAR_TABLE, which handles tables of states, transitions, labels, etc.

  A number of tools have been developed within the OPEN/CÆSAR environment, among which:
  - BISIMULATOR, which checks bisimulation equivalences and preorders,
  - DETERMINATOR, which eliminates nondeterminism in normal, probabilistic, or stochastic systems,
  - DISTRIBUTOR, which generates the graph of reachable states using several machines,
  - EVALUATOR, which evaluates regular alternation-free μ-calculus formulas,
  - EXECUTOR, which performs random execution,
  - EXHIBITOR, which searches for execution sequences matching a given regular expression,
  - GENERATOR, which constructs the graph of reachable states,
  - PROJECTOR, which computes abstractions of communicating systems,
  - REDUCTOR, which constructs and minimizes the graph of reachable states modulo various equivalence relations,
  - SIMULATOR, XSIMULATOR, and OCIS, which allow interactive simulation, and
  - TERMINATOR, which searches for deadlock states.
- BCG (*Binary Coded Graphs*) is both a file format for storing very large graphs on disk (using efficient compression techniques) and a software environment for handling this format. BCG also plays a key role in CADP as many tools rely on this format for their inputs/outputs. The BCG environment consists of various libraries with their programming interfaces, and of several tools, such as:
  - BCG_DRAW, which builds a two-dimensional view of a graph,
  - BCG_EDIT, which allows to modify interactively the graph layout produced by BCG_DRAW,
  - BCG_GRAPH, which generates various forms of practically useful graphs,
  - BCG_INFO, which displays various statistical information about a graph,
  - BCG_IO, which performs conversions between BCG and many other graph formats,

  - BCG_LABELS, which hides and/or renames (using regular expressions) the transition labels of a graph,
  - BCG_MERGE, which gathers graph fragments obtained from distributed graph construction,
  - BCG_MIN, which minimizes a graph modulo strong or branching equivalences (and can also deal with probabilistic and stochastic systems),
  - BCG_STEADY, which performs steady-state numerical analysis of (extended) continuous-time Markov chains,
  - BCG_TRANSIENT, which performs transient numerical analysis of (extended) continuous-time Markov chains, and
  - XTL (*eXecutable Temporal Language*), which is a high level, functional language for programming exploration algorithms on BCG graphs. XTL provides primitives to handle states, transitions, labels, *successor* and *predecessor* functions, etc. For instance, one can define recursive functions on sets of states, which allow to specify in XTL evaluation and diagnostic generation fixed point algorithms for usual temporal logics (such as HML [55], CTL [51], ACTL [52], etc.).

- The connection between explicit models (such as BCG graphs) and implicit models (explored on the fly) is ensured by OPEN/CÆSAR-compliant compilers, e.g.:
  - CÆSAR.OPEN, for models expressed as LOTOS descriptions,
  - BCG_OPEN, for models represented as BCG graphs,
  - EXP.OPEN, for models expressed as communicating automata, and
  - SEQ.OPEN, for models represented as sets of execution traces.

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

The CADP tools are well-integrated and can be accessed easily using either the EUCALYPTUS graphical interface or the SVL [5] scripting language. Both EUCALYPTUS and SVL provide users with an easy, uniform access to the CADP tools by performing file format conversions automatically whenever needed and by supplying appropriate command-line options as the tools are invoked.

4.2. The TRAIAN Compiler

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

We develop a compiler named TRAIAN for translating descriptions written in the LOTOS NT language (see § 2.3) into C programs, which will be used for simulation, rapid prototyping, verification, and testing.

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

Although this version of TRAIAN is still incomplete (it does not handle LOTOS NT processes), it already has useful applications in compiler construction [7]. The recent compilers developed by the VASY project team — namely AAL, CHP2LOTOS (see § 5.2.3), EVALUATOR 4.0, EXP.OPEN 2.0 (see § 5.1.5), FSP2LOTOS (see § 5.2.3), LNT2LOTOS (see § 5.2.2), NTIF (see § 2.3), and SVL (see § 5.1.5) — all contain a large amount of LOTOS NT code, which is then translated into C code by TRAIAN.

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

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

5. New Results

5.1. Models and Verification Techniques

5.1.1. The CÆSAR_SOLVE Library

Participant: Radu Mateescu.

CÆSAR_SOLVE is a generic software library for solving boolean equation systems of alternation depth 1 (i.e., without mutual recursion between minimal and maximal fixed point equations) on the fly. This library is at the core of several CADP verification tools, namely the equivalence checker BISIMULATOR (see § 5.1.2), the model checker EVALUATOR 3.5 (see § 5.1.3), and the minimization tool REDUCTOR 5.0 (see § 5.1.4). The resolution method is based on boolean graphs, which provide an intuitive representation of dependencies between boolean variables, and which are handled implicitly, in a way similar to the OPEN/CÆSAR interface [3].

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

In 2006, the CÆSAR_SOLVE library (12, 200 lines of C code) was improved as follows:

- The primitive for writing a boolean equation system to a text file was enhanced in order to write not only the whole system, but also the portion of the system representing the diagnostic produced after solving a given boolean variable.
- The primitive for reading a boolean equation system from a text file was enhanced in order to handle the cases where the equation blocks and the boolean variables in the left-hand sides of the equations of a block are numbered neither contiguously, nor increasingly. This allows to read text files containing diagnostics of resolutions, which do not necessarily fulfill these two conditions.
- The primitives for reading and writing a boolean equation system from/to a text file were enhanced in order to support on the fly compression, which can reduce the size of text files by several orders of magnitude. This possibility is exploited by BISIMULATOR (see § 5.1.2) and EVALUATOR (see § 5.1.3).
- The A4 algorithm was enhanced to detect cycles of the boolean graph that contain certain boolean variables marked by a predicate provided by the user application. This feature is useful for encoding the evaluation of certain temporal logic properties describing infinite, unfair execution sequences. Also, a bug was corrected in algorithm A4 when detecting disjunctive/conjunctive boolean graphs.

An article about CÆSAR_SOLVE was published in an international journal [22].

5.1.2. The BISIMULATOR Tool

Participant: Radu Mateescu.

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

BISIMULATOR works by reformulating the graph comparison problem in terms of a boolean equation system, which is solved on the fly using the CÆSAR_SOLVE library (see § 5.1.1). A useful functionality of BISIMULATOR is the generation of a "negative" diagnostic (i.e., a counterexample), which explains why two graphs are not equivalent (or not included one in the other). The diagnostics generated by BISIMULATOR are directed acyclic graphs and are usually much smaller than those generated by other tools (such as ALDÉBARAN) that can only generate counterexamples restricted to sets of traces.

In 2006, we continued the development of BISIMULATOR ($15,900$ lines of C code). In addition to a bug fix related to the counterexample generation for branching equivalence:

- A new command-line option was added to BISIMULATOR to apply τ-confluence reduction on the implicit graph when comparing modulo branching or observational equivalence. When the implicit graph contains interleavings due to the presence of loosely-coupled parallel processes, this option can reduce the time and memory required for the verification by up to one order of magnitude.
- The encoding of observational equivalence in terms of boolean equation systems was enhanced in order to simplify the equations when the explicit graph is deterministic and does not contain τ-transitions. In this case, observational equivalence becomes identical to $\tau^*.a$ equivalence, except for the states of the implicit graph from which a deadlock state can be reached after zero or more τ-transitions. These states are now detected during the computation of τ-closures (transitive reflexive closures over τ-transitions) and used to simplify the equations accordingly. This can reduce the number of boolean variables by up to 30%.

5.1.3. The EVALUATOR Tool

Participants: Radu Mateescu, Damien Thivolle.

EVALUATOR is a model checker that evaluates a temporal logic property on a graph represented implicitly using the OPEN/CÆSAR environment. Properties are described in regular alternation-free μ-calculus, a logic built from boolean operators, possibility and necessity modalities containing regular expressions denoting transition sequences, and fixed point operators without mutual recursion between least and greatest fixed points. The input language of the tool also allows to define parameterized temporal operators and to group them into separate libraries.

EVALUATOR works on the fly, meaning that only those parts of the implicit graph pertinent to verification are explored. The model checking problem is reformulated in terms of solving a boolean equation system. A useful feature of EVALUATOR is the generation of diagnostics (examples and counterexamples) explaining why a formula is true or false.

In 2006, we continued the development of the EVALUATOR 3.5 tool. In particular, the translation in regular alternation-free μ-calculus of the inevitability operator of ACTL was improved. When using EVALUATOR 3.5 to check temporal formulas containing this operator, the new translation leads to gains in time and memory up to a factor 8.

We also continued our work (undertaken in 2003) for extending the regular alternation-free μ-calculus with new operators dedicated to the specification of properties involving data values. This led to a prototype EVALUATOR 4.0 ($37,600$ lines of SYNTAX/LOTOS NT code and $11,100$ lines of C code), which brings the following enhancements with respect to EVALUATOR 3.5:

- State formulas are extended with data-handling operators inspired from programming languages, such as "**if-then-else**" and "**case**". Fixed point operators are enhanced with data parameters allowing arbitrary calculations to be performed on the fly while exploring the graph. Action formulas are extended with action patterns that extract the data values contained in transition labels and store them in variables that can be referred to in the formula. Regular expressions occurring inside modalities are extended with iteration operators ranging over natural intervals, and also with programming language constructs such as "**while**", "**until**", and "**for**". Finally, special operators are introduced

for capturing states of the graph and manipulating them in formulas; this allows to express non-standard properties, such as the existence of self-loops (transitions from a state to itself) and past-time properties (occurence of actions before a certain state). This new language of formulas is called MCL (*Model Checking Language*). It supersedes the regular alternation-free μ-calculus accepted as input by EVALUATOR 3.5.

- The problem of evaluating an MCL formula on the fly amounts to the local resolution of a boolean equation system, which is performed using the CÆSAR_SOLVE library. The translation consists of several phases: lexical, syntactic, and semantic analysis of the MCL formula; type checking and replacement of the derived operators by primitive ones; conversion to positive normal form by propagating negations down to the atomic formulas; generation of modal equation systems containing parameterized propositional variables in the left-hand sides and modal formulas in the right-hand sides; elimination of regular expressions by translating them into terms of modalities and fixed point equations. Then, the resulting modal equation system and the graph (represented implicitly using the OPEN/CÆSAR environment) are combined together into a boolean equation system (represented implicitly according to the CÆSAR_SOLVE interface) containing a distinguished variable, whose local resolution yields the truth value of the MCL formula on the initial state of the graph.
- The translation of a state formula into a modal equation system was optimized in order to maximize the number of blocks in the final boolean equation system whose corresponding boolean graphs are disjunctive/conjunctive. These blocks are solved using the memory-efficient algorithm A4 of CÆSAR_SOLVE, which avoids storing the transitions of the boolean graphs (and hence the transitions of the graph on which the formula is evaluated). In practice, all temporal formulas built from the operators of CTL, ACTL, and PDL are translated into boolean equation systems containing only disjunctive/conjunctive blocks, and are therefore evaluated with a quasi-optimal memory consumption using A4.

The prototype version EVALUATOR 4.0 was successfully tested on $2,300$ examples of MCL formulas and on all regular alternation-free μ-calculus formulas available in the demo examples of the CADP distribution.

5.1.4. The REDUCTOR and DETERMINATOR Tools

Participants: Frédéric Lang, Radu Mateescu.

The REDUCTOR 4.0 tool of CADP, developed in 2005, implements several forms of (partial or total) graph reductions. Some of these reductions are obtained by encoding the reduction problem into a boolean equation system that is solved on the fly using the CÆSAR_SOLVE library (see § 5.1.1).

CADP also contains a tool named DETERMINATOR, which eliminates nondeterminism from ordinary or stochastic graphs.

In 2006, we released a new version 5.0 of REDUCTOR, together with a new version of DETERMINATOR. REDUCTOR 5.0 includes several functionalities previously available in DETERMINATOR. The main changes are the following:

- We fixed a bug in reduction modulo safety equivalence, which caused the generated labeled transition system to be not minimal in some cases.
- Three new equivalences were added to REDUCTOR, namely trace reduction (previously available in DETERMINATOR as normal determinization), weak trace reduction (previously available in DETERMINATOR as determinization with τ-elimination), and τ divergence reduction.

5.1.5. Compositional Verification Tools

Participants: Hubert Garavel, Frédéric Lang.

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

PROJECTOR 2.0 implements behavior abstraction [54], [59] by taking into account interface constraints.

EXP.OPEN 2.0 explores on the fly the graph corresponding to a network of communicating automata (represented as a set of BCG files).

SVL (*Script Verification Language*) is both a high level language for expressing complex verification scenarios and a compiler dedicated to this language.

In 2006, we enhanced these tools along the following lines:

- We corrected one bug in PROJECTOR 2.0, two bugs in EXP.OPEN 2.0, and four bugs in SVL.
- We enhanced EXP.OPEN 2.0 to convert networks of communicating automata into Petri nets encoded in the TPN format of the TINA toolbox developed at LAAS-CNRS.
- We added to EXP.OPEN 2.0 a new operator for specifying priorities between the transitions of a network of communicating automata.
- We extended the SVL language to support the new equivalences available in REDUCTOR 5.0 (see § 5.1.4) and the new features of BISIMULATOR (see § 5.1.2) and EVALUATOR (see § 5.1.3), e.g., selection between depth-first or breadth-first search algorithms, selection of algorithms dedicated to acyclic graphs, etc.

An article on refined interface generation using EXP.OPEN and SVL (see the VASY 2005 activity report) was published in an international conference [29].

5.1.6. Parallel and Distributed Verification Tools

Participants: Jérôme Fereyre, Hubert Garavel, Radu Mateescu.

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

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

DISTRIBUTOR 3.0 and BCG_MERGE 3.0 are now properly documented and integrated into CADP. A tool paper was published at an international conference [27].

In 2006, this work progressed as follows:

- We studied various enhancements of our CÆSAR_NETWORK library, which implements generic communication primitives for distributed verification tools. We simplified its programming interface by reducing the number of primitives needed to initialize and terminate a distributed computing session. We also designed an extension of the GCF (*Grid Configuration File*) format used by CÆSAR_NETWORK in order to support the mainstream job schedulers available in clusters and grids.
- Each graph generated by executing DISTRIBUTOR on a set of machines is represented as a PBG (*Partitioned* BCG *Graph*) file, which consists of a set of graph fragments generated on each machine and stored as BCG files. The PBG format provides various information for handling these fragments (number of states and transitions of each fragment, GCF file used to generate the fragments, log files produced on each machine, etc.). So far, the only tool of CADP handling the PBG format was BCG_MERGE, which translates a PBG file into a BCG file by merging all fragments into a single graph. In 2006, we developed four new prototype tools operating on PBG files:
  - PBG_CP copies a PBG file and its dependencies (fragments, log files, and GCF file) from a machine to another,

- PBG_MV moves a PBG file (and its dependencies) between two machines,
- PBG_RM removes a PBG file (and its dependencies), and
- PBG_OPEN provides a distributed algorithm that implements the OPEN/CÆSAR programming interface [3], thus allowing to explore on the fly a PBG file (without merging its fragments first as BCG_MERGE does).

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

In 2006, we continued the development of this library. The library code (16,000 lines of C code) was thoroughly reviewed and simplified. The error management was improved. The primitive for writing a boolean equation system (or the portion of the system corresponding to the diagnostic of a given variable) to a file was also implemented in the distributed version of CÆSAR_SOLVE. The distributed resolution algorithm for boolean equation systems was enhanced to detect on the fly cyclic dependencies between equation blocks (the presence of such dependencies indicates that the boolean equation system has an alternation depth greater than one).

An article about the distributed version of CÆSAR_SOLVE and its applications was published in an international conference [28].

We developed a new tool, named BES_SOLVE, which supersedes two existing prototype tools for generating random boolean equation systems and for reading and writing boolean equation systems from/to text files, respectively. BES_SOLVE allows to compare and cross-check the various resolution algorithms provided by the sequential and distributed versions of CÆSAR_SOLVE. It constructs a boolean equation system in memory either by reading it from a (possibly compressed) text file, or by generating it randomly according to various parameters (number of equation blocks, minimum and maximum number of variables in each block, length of the equations, percentage of boolean constants, percentage of disjunctive and conjunctive variables in the right-hand sides of the equations, seed value for initializing the random number generator, etc.). Then, a boolean variable defined in some equation block of the boolean system can be solved by invoking any sequential or distributed algorithm of CÆSAR_SOLVE. BES_SOLVE served to experiment intensively the resolution algorithms and allowed to correct a bug in algorithm A4 of the sequential version of CÆSAR_SOLVE.

5.1.7. Other Tool Developments

Participants: David Champelovier, Jérôme Fereyre, Hubert Garavel, Frédéric Lang, Nathalie Lépy, Radu Mateescu, Wendelin Serwe, Marie Vidal.

We undertook the design of an integrated development environment for CADP within the ECLIPSE framework. This environment comprises both a LOTOS editor and a graphical user-interface (inspired from EUCALYPTUS but based on ECLIPSE) for the CADP tools.

Additionally, we improved the following CADP tools and libraries:

- We fixed one bug in the CAESAR_REGEXP library and two bugs in XTL.
- We enhanced DETERMINATOR and BCG_MIN to support double precision floating point numbers.
- We improved the EUCALYPTUS graphical user-interface to provide access to all recent features and tools of CADP.
- We added a new CADP demo example (WEB services for stock management and an on-line book auction) and we updated most of the other demo examples to take advantage of the most recent features and tools of CADP.

We continued adapting CADP to the latest computing platforms:

- We finished porting CADP to recent LINUX distributions and MAC OS 10.4 "TIGER".
- We modified CADP to support recent C compilers (GCC 4, INTEL ICC 9.0, SUN STUDIO 11).
- We improved the portability of CADP by adding new "wrappers", i.e., shell scripts that make CADP less dependent on the underlying operating system.
- As regards 64-bit architectures:
  - We wrote a portability guide for 64-bit applications.
  - We started porting the shell scripts of CADP as well as third party software used by CADP (e.g., CUDD, SPARSE, TCL-TK/TIX, garbage collector) to 64-bit architectures.

5.2. Languages and Compilation Techniques

5.2.1. Compilation of LOTOS

Participants: Hubert Garavel, Wendelin Serwe.

The CADP toolbox contains several tools dedicated to the LOTOS language, namely: the CÆSAR.ADT compiler [2] for the data type part of LOTOS, the CÆSAR compiler [9] for the process part of LOTOS, and the CÆSAR.INDENT pretty-printer.

In 2006, we performed maintenance activities for these tools (two bugs fixed in CÆSAR, one bug fixed in the CÆSAR.BDD tool invoked by CÆSAR, and two enhancements in CÆSAR.INDENT). We also improved the C code generated by CÆSAR and CÆSAR.ADT to avoid warnings emitted by the most recent C compilers.

We pursued our study of state space reduction techniques, our goal being to decrease the size of the graphs generated by CÆSAR, still preserving strong bisimulation between the original and reduced graphs.

Our work on state space reduction based on live variable analysis resulted in version 7.0 of CÆSAR (previously named CÆSAR.NEW), which was officially released as part of CADP in July 2006. On all CADP demos, CÆSAR 7.0 reduced the state space by a mean factor of 45 (we observed a maximum factor of 4,400) as regards the number of states and by a mean factor of 38 (we observed a maximum factor of 3,100) as regards the number of transitions. This work led to a journal publication [20].

Additionally, W. Serwe experimented further uses of data-flow analysis to improve the efficiency for enumerative verification. A prototype version of CÆSAR was developed and experimented in the framework of the FORMALFAME PLUS contract (see § 6.1): we obtained a memory reduction by a factor of 1.4 and a time reduction by a factor of 2.

5.2.2. Compilation of E-LOTOS and LOTOS NT

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

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

- The TRAIAN compiler (see § 4.2) remained stable in 2006, but we developed a new tool, named TRAIAN.INDENT, for indenting LOTOS NT programs, similar to the existing tool CÆSAR.INDENT (see § 5.2.1).
- In the framework of the FORMALFAME PLUS contract (see § 6.1), we continued the development of a tool suite for the translation from LOTOS NT to LOTOS, which aims at easing the development of large specifications by BULL and to reuse the existing LOTOS tools for analyzing concurrent systems described in LOTOS NT. The tool suite consists of a LOTOS/LOTOS NT preprocessing tool named LPP, a translator from LOTOS NT data types to LOTOS named LNT2LOTOS, and a shell script named LNT_COMPILE, which calls LPP and LNT2LOTOS.

In 2006, the tool suite was improved as follows:

- The predefined comparison functions ("$=$", "$<$", "$\leq$", etc.), which were only available for enumerated types in 2005, were made available to all constructor types, using a recursive lexicographic ordering of constructors and constructor fields.
- A new predefined type "**sorted list of** T" was added. The elements of a sorted list are sorted automatically using the order relation "$<$" on T, which can be either generated automatically as explained in the previous item, or given by the user.
- We implemented the translation from LOTOS NT functions into LOTOS equations, starting from an algorithm [64] that translates into Horn clauses a subset of the C language ("**while**" loops without "**break**" statements and functions with value passing parameters only and a single "**return**" statement located at the end of the function). We extended this algorithm so as to handle reference passing parameters, pattern matching ("**case**" statement), loop interruptions ("**break**" statement), multiple "**return**" statements within the body of functions, uncatchable exceptions ("**raise**" statement), and function name overloading.
- The LNT_COMPILE tool was improved to allow the generated LOTOS code to be combined with handwritten LOTOS code and/or C code provided by the user.

LNT2LOTOS is developed using the SYNTAX/TRAIAN technology [7]. It grew from $3,660$ lines (760 lines of SYNTAX code, $1,920$ lines of LOTOS NT code, and 980 lines of C code) at the end of 2005 up to $17,300$ lines ($2,100$ lines of SYNTAX code, $14,000$ lines of LOTOS NT code, and $1,200$ lines of C code) at the end of 2006.

In 2006, we delivered 7 successive versions of the tool suite to BULL, who uses LOTOS NT to model a critical part of its FAME2 multiprocessor architecture for high-end servers (see § 6.1). A non-regression test suite of 67 programs representing more than $6,000$ lines of LOTOS NT code was developed. The reference manual was updated [35] and grew from 29 pages (at the end of 2005) up to 47 pages (at the end of 2006). A forge was set up under INRIA GFORGE to track bugs and feature requests, and to serve as a repository where our BULL partners can download the new versions of the LNT2LOTOS tool suite.

5.2.3. Source-Level Translations between Concurrent Languages

Participants: Hubert Garavel, Abdul Malik Khan, Frédéric Lang, Olivier Ponsini, Gwen Salaün, Wendelin Serwe.

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

To address this problem, we started investigating source-level translators from various process algebras into LOTOS, so as to widen the applicability of the CADP tools. One first example was the LNT2LOTOS tool suite (see § 5.2.2). In 2006, we studied also translators for other concurrent languages:

- We considered the process algebra FSP (*Finite State Processes*) defined in a popular textbook on concurrency [60]. Continuing the collaboration initiated in 2005 with Jeff Kramer and Jeff Magee (Imperial College, London, see § 8.3), we removed the ambiguities found in the reference FSP grammar and provided a new unambiguous LALR(1) grammar for FSP. We redesigned our FSP2LOTOS prototype, undertaken in 2005, that translated the "basic FSP" fragment (i.e., FSP without its data part and without syntactic sugar) to LOTOS. The new FSP2LOTOS prototype ($5,000$ lines of SYNTAX code, $20,000$ lines of LOTOS NT code, and 500 lines of C code) translates "full FSP" into LOTOS. It is available on SOLARIS, LINUX, and WINDOWS and has been tested on $10,500$

lines of FSP code, including many examples given in the FSP textbook [60].

- In the framework of the FIACRE (see § 7.1), OPENEMBEDD (see § 6.3), and TOPCASED (see § 6.4) projects, and in cooperation with the LAAS-CNRS and IRIT laboratories, we undertook the definition of a new intermediate model named FIACRE (*Format Intermédiaire pour les Architectures de Composants Répartis Embarqués*). Derived from NTIF [6] and V-COTRE [45], FIACRE will be used as a pivot formalism between modeling languages (such as AADL, UML, or SYSML) and verification tools (such as CADP and TINA). After several meetings, intensive technical correspondence (80 e-mail exchanges), and 8 drafts, we converged to a 14-page document describing FIACRE.

- In the framework of the INRIA/LETI collaboration (see § 7.1), we focused on the process algebra CHP (*Communicating Hardware Processes*) for which the TIMA laboratory has developed a circuit synthesis tool named TAST [65] and which is used by the LETI laboratory to describe complex, asynchronous circuits at a high abstraction level. The goal is to integrate formal verification into the design flow of complex microelectronic circuits.

  In 2006, we improved the CHP2LOTOS translator by developing code specialisation techniques that optimize the translation of each channel depending on its profile (i.e., whether and how the CHP "probe" operator is applied to this channel). Computing channel profiles in a pre-processing step and optimizing the translation accordingly allows to reduce the state space significantly: on the example of the FAUST circuit, an asynchronous NOC (*Network on Chip*) developed at the LETI laboratory (see § 5.3), we observed a reduction of the number of states (respectively, transitions) by a factor of 89 (respectively, 156).

  Our CHP2LOTOS translator (currently, $2,200$ lines of SYNTAX code, $13,400$ lines of LOTOS NT code, and $3,900$ lines of C code) was extended accordingly. We also added more examples to the test base (currently, about 500 CHP specifications, corresponding to $14,500$ lines of CHP code). CHP2LOTOS has been tested on SOLARIS, LINUX, and WINDOWS.

- We also started to investigate the verification of TLM (*Transaction Level Model*) specifications. Compared to traditional RTL (*Register Transfer Level*) models, TLM models are more suitable for faster simulation, simultaneous development of software and hardware, and earlier hardware/software partitioning.

  Among all TLM languages, SYSTEMC [57] emerges as an industrial standard. SYSTEMC is a C++ library providing both a high-level description language and a simulation kernel. However, integrating formal verification with a system design flow based on SYSTEMC/TLM is still an open question as the process scheduler of the SYSTEMC simulation kernel may differ from the actual hardware behavior.

  In this perspective, we are currently investigating the translation of (a TLM subset of) SYSTEMC into LOTOS or LOTOS NT (see § 5.2.2). For this purpose, we studied the SYSTEMC front-end PINAPA [61], which we ported to the SOLARIS operating system.

- In collaboration with Gregor Goessler (POPART project), we designed a translator that connects to CADP the PROMETHEUS tool developed by POPART. This translator [36] takes as input a model in the BIP format of PROMETHEUS and performs four main tasks:

  1. It converts each sequential component of the BIP behavior layer (a typical condition/action model) into a sequential LOTOS process that will be translated automatically into a labeled transition system (BCG file) using CADP.
  2. It converts the BIP interaction layer into an EXP.OPEN file, which composes the BCG graphs using synchronization vectors.
  3. It takes into account the global constraints expressed in the BIP execution layer either as invariants, or as interaction constraints that determine which transition can be fired (a special case of interaction constraints being priorities between transitions).

Such constraints impact the generation of LOTOS and EXP.OPEN described in items 1. and 2. above in three ways. First, special actions, called observers, are added in LOTOS sequential processes to enable observation of local variables occurring in state invariants and/or interaction constraints. Second, synchronization vectors between observers are added to the EXP.OPEN model to identify all states in which the invariants or interaction constraints are violated. Third, priorities are added to the EXP.OPEN model to cut each transition that violates some interaction constraints, or whose source state violates some invariant.

4. It generates an SVL script that manages the generation of the BCG files described in item 2 above. This script can then be extended to explore the state space of the input BIP model on the fly using EXP.OPEN and other CADP tools.

This translator was integrated as a new module in the PROMETHEUS tool. It was experimented on a hardware architecture.

5.3. Case Studies and Practical Applications

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

In 2006, the VASY project team also worked on the following case studies:

- We continued our collaboration with Antonella Chirichiello (University "La Sapienza", Rome) on the use of LOTOS and CADP to design and verify WEB services. This led to a new publication [26] describing an e-business application specified in LOTOS, verified with CADP, and translated into the standard orchestration language BPEL.
- In the context of the SENVA collaboration (see § 7.2), we continued the study (undertaken in 2005) of a turntable system for drilling products. This industrial critical system is interesting because its distributed embedded controller is simple enough to have a concise formal description, but sufficiently complex to require formal analysis. This system was previously specified by CWI and the University of Eindhoven using several languages (χ, μCRL, PROMELA, and timed automata) and analyzed with various tools [46].

  We formally specified in LOTOS a sequential and a distributed version of the controller embedded in the turntable system, and we formulated in regular alternation-free μ-calculus a set of safety and liveness properties characterizing its correct behavior. Using the BISIMULATOR (see § 5.1.2) and EVALUATOR 3.5 (see § 5.1.3) tools of CADP, we checked the compatibility between both versions of the controller and their correctness with respect to the temporal properties. This activity led to the publication of a book chapter [23].
- In the context of the FORMALFAME PLUS contract (see § 6.1), we worked on a critical part of BULL's FAME2 multiprocessor architecture, the PAB (*Pipeline and Active transaction Block*), for which BULL wrote a LOTOS NT specification (3,977 lines of LOTOS NT) that was translated to LOTOS (5,145 lines of LOTOS) using our LNT2LOTOS tool suite (see § 5.2.2).

  This specification was used for two purposes. On the one hand, it allowed to generate execution traces that were used to test the VERILOG code of the PAB; this allowed BULL to detect coding errors. On the other hand, it was used to verify the correctness of the PAB protocol itself. Confronted to state explosion issues, we performed several experiments:

  – We first optimized the LOTOS specification, taking into account the symmetries induced by lists containing the same elements in different order. We introduced sorted lists instead of simple lists and provided external C code to reduce memory usage. This divided the state space size by a factor of 2.4 (from 900,000 states down to 379,000 states), the generation time by 37 (from 1 hour 19 minutes to 2 minutes 8 seconds), and the memory usage by 6.3 (from 189 down to 30 MBytes).

  - Using the DISTRIBUTOR tool running on the GRIMAGE cluster of INRIA Rhône-Alpes (36 processors: 10 bi-OPTERON at 2 GHz and 8 bi-XEON at 3 GHz with 1 GByte RAM each), we were able to generate a state space of 10 million states and 14 million transitions in 2 minutes.
  - Finally, BULL managed to split the PAB specification in two independent parts, replacing the generation of one single, large graph by the generation of two separate, smaller graphs. BULL also simplified the specification based on symmetry arguments. After these simplifications, BULL could generate the graph on a single sequential machine in a few hours.

- In the context of the INRIA/LETI collaboration (see § 7.1), we pursued the study (undertaken in 2005) of the FAUST NOC (*Network on Chip*) circuit developed by the LETI laboratory.

  In 2006, we focused on the communication interconnect part of FAUST [41], which routes packets (consisting of several 34-bit flits) between the 23 components of the FAUST circuit. This interconnect is described in the hardware process calculus CHP (*Communicating Hardware Processes*) and implemented, at the RTL level, in asynchronous logic. The interconnect has 23 communication nodes, one per component. Each communication node consists of five input and five output controllers. Each input controller dispatches incoming flits to one out of four output controllers, and each output controller arbitrates between four input controllers.

  To carry out the compositional verification of an input controller, we used the following steps:

  - First, applying the idea of data independence, we reduced the potential state space from 10^{25} to 5.10^{16} states by setting parts of the flits to a fixed bit pattern. Then, we further reduced the state space to a manageable size by considering several scenarios (sequences of 4 flits) carefully chosen according to the properties to be verified (e.g., data integrity and correct routing of flits/packets). These two reductions were obtained by introducing additional CHP processes, called "traffic generators", which restrict the environment of the input controller by providing meaningful inputs only. Applying our CHP2LOTOS translator (see § 5.2.3) to the CHP description of the input controller (500 lines of CHP code) connected to each traffic generator (about 700 lines of CHP code), we produced a set of LOTOS specifications in less than one second.
  - Second, for each LOTOS specification, we generated the corresponding state space using the compositional verification techniques of CADP (see § 5.1.5) using a generic SVL script (41 steps, 450 lines of SVL code); for a typical scenario, the generated state space had 1,300 states and 3,116 transitions, and the largest intermediate state space had 295,893 states and 812,283 transitions.
  - Third, we used the equivalence checking and model checking tools of CADP to verify seven properties, such as absence of deadlocks, correctness of the communication protocol, integrity of the transmitted data, and correctness of flit/packet routing. These verification steps were automated using an SVL script (250 lines of SVL code).

  We were able to exhibit a routing error in the CHP description. Although this error had been already found (and fixed) manually during the synthesis of the FAUST circuit (it required more than 500,000 steps to replay the error in the TAST native simulator for CHP), our approach based on CHP2LOTOS and CADP allowed to detect the error automatically in less than 15 minutes.

  This work led to a joint INRIA/LETI publication [34].

- We continued our collaboration with Estelle Dumas, Hidde de Jong, and Delphine Ropers (HELIX project team of INRIA Rhône-Alpes) for connecting CADP and the GNA (*Genetic Network Analyzer*) tool developed by HELIX in order to verify temporal properties of genetic regulatory networks.

GNA provides a simulator of qualitative models of genetic regulatory networks in the form of piecewise-linear differential equations. The output of the simulator is a Kripke structure (i.e., a state-transition graph in which the relevant information is associated to states) that can also be exported by GNA as a labeled transition system, thus enabling a direct connection with CADP. This allows to enhance GNA with verification features.

In practice, GNA is used mainly by biologists, who are not necessarily familiar with computer science and formal verification. For this community, we are studying a transparent connection between CADP and GNA based on a client-server architecture, in which CADP is installed on a single server and several instances of GNA are running on remote client machines. A client sends Kripke structure files and temporal properties to the server and gets back verdicts and diagnostics. A prototype connection based on WEB services was developed and experimented on several biologically-relevant examples.

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

- the verification of the Transport Layer Security protocol [49],
- the test case generation using mutation-based testing techniques [53], [37],
- the verification of fault-tolerant ERLANG programs [43], [44],
- the solving of scheduling problems using untimed model checking [67], [68],
- the verification of software components [38], [39],
- the formal analysis of an automatic document feeder [63],
- the formal analysis and verification of a digital rights management protocol [58],
- the verification of privacy using observational determinism [56],
- the performability analysis of the European Train Control System [48],
- the verification of a WIFI Internet access system available in airports [40].

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:

- an environment for the verification of the VODKA video-on-demand system [66],
- the ADAPTOR tool, developed at the IBISC laboratory (Evry), which allows the automatic generation of adaptors between differing component interfaces,
- the VESTA tool [42], [62], developed at the LIFC laboratory (Besançon), which allows the verification of divergence-sensitive and stability respecting τ-simulation for component-based timed systems,
- the ANNOTATOR tool [69], developed at the university of Málaga (Spain), for the static analysis of software,
- a framework for model checking socket-based concurrent C programs [70], developed at the university of Málaga (Spain),
- the VERCORS platform [40] for model checking distributed components, developed by the OASIS project team at INRIA Sophia-Antipolis.

Finally, a textbook [47] was published, which uses CADP as a software support for teaching concurrency theory.

6. Contracts and Grants with Industry

6.1. The FormalFame Plus Contract

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

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

Between 1995 and 1998, two case studies were successfully tackled using CADP: the POWERSCALE bus arbitration protocol [50] and the POLYKID multiprocessor architecture [12].

Between 1998 and 2004, the collaboration focused on FAME, the CC-NUMA multiprocessor architecture used in BULL's NOVASCALE series of high-performance servers based on INTEL ITANIUM processors. The CADP tools have been used to validate a crucial circuit of FAME – the FSS (*Fame Scalability Switch*) – that implements the cache coherency protocol.

In 2004, the collaboration was renewed by a followup contract named FORMALFAME PLUS, which, in 2005, was extended for two more years. FORMALFAME PLUS aims at enhancing the performance and usability of the CADP tools to address the FAME2 multiprocessor architecture under design at BULL for their future high-end servers.

In 2006, the contributions of VASY to FORMALFAME PLUS were the following:

- We continued the development of our LNT2LOTOS tool suite (see § 5.2.2), which was used by BULL.
- We experimented various abstraction and verification techniques on a critical part of BULL's FAME2 multiprocessor architecture (see § 5.3).

The FORMALFAME PLUS contract will find its continuation in the MULTIVAL project (see § 6.2).

6.2. The Multival Project

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

MULTIVAL (*Validation of Multiprocessor Multithreaded Architectures*) is a project of MINALOGIC, the French *pôle de compétitivité* dedicated to micro-nano technologies and embedded software for systems on chip (EMSOC cluster). MULTIVAL addresses verification and performance evaluation issues for three innovative asynchronous architectures developed by BULL, CEA/LETI, and ST MICROELECTRONICS.

In 2006, MULTIVAL was approved for joint funding by the French government (*Fonds de compétitivité des entreprises*) and *Conseil général de l'Isère*. MULTIVAL started in December 2006 for three years.

6.3. The OpenEmbeDD Project

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

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

OPENEMBEDD started in May 2006 for three years. In 2006, our contributions focused on the identification of a model-based architecture for CADP and the definition of the FIACRE intermediate model for embedded systems (see § 5.2.3).

6.4. The Topcased Project

Participants: Hubert Garavel, Frédéric Lang, Nathalie Lépy, Jan Stoecker.

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

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

TOPCASED started in August 2006 for four years. In 2006, we worked along the following lines:

- We contributed actively to the AIRBUS proposal for adding a behavioral annex to AADL (*Architecture Analysis & Design Language*), the SAE (*Society of Automotive Engineers*) standard architecture description language for embedded real-time systems. The behavioral annex provides structured statements inspired from NTIF [6] that describe data computations using a "big steps" semantics, which avoids splitting these computations into many smaller, less efficient steps. The behavioral annex will be submitted for balloting in early 2007.
- We undertook the design of an integrated development environment for CADP based on ECLIPSE (see § 5.1.7).
- We started a study of time models so as to identify time models which are both equipped with effective verification algorithms and appropriate to describe asynchronous systems with data and time.

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.

6.5. Forthcoming Projects

Participants: Hubert Garavel, Radu Mateescu.

In 2006, VASY contributed to the submission of the EC-MOAN (*Scalable modeling and analysis techniques to study emergent cell behavior: Understanding the E. coli stress response*) proposal, which was accepted for funding within the NEST PATHFINDER European program. EC-MOAN aims at the development of new, scalable methods for modeling and analyzing integrated genetic, metabolic, and signaling networks, and the application of these methods for a better understanding of a bacterial model system. EC-MOAN will start in February 2007 for three years.

7. Other Grants and Activities

7.1. National Collaborations

The VASY project team plays an active role in the joint research center launched in 2004 between INRIA Rhône-Alpes and the LETI laboratory of CEA-Grenoble. In co-operation with LETI scientists (Edith Beigné, François Bertrand, Fabien Clermidy, Yvain Thonnart, and Pascal Vivet), VASY develops software tools for the design of asynchronous circuits and architectures such as GALS (*Globally Asynchronous Locally Synchronous*), NOCS (*Networks on Chip*), and SOCS (*Systems on Chip*). In 2006, our work focused on the CHP2LOTOS translator (see § 5.2.3) and its application to the FAUST architecture (see § 5.3).

Together with the OASIS project team of INRIA Sophia-Antipolis (Antonio Cansado and Eric Madelaine), the LTCI team of ENST-Paris (Irfan Hamid, Elie Najm, and Sylvie Vignes), the SVF team of the LAAS-CNRS laboratory (Bernard Berthomieu, Florent Peres, and François Vernadat), and the MVR team of IRIT (Mamoun Filali), VASY participates to the national action FIACRE (ACI *Sécurité Informatique*) started in 2004 (see http://www-sop.inria.fr/oasis/fiacre). In 2006, we implemented new interconnections (see § 5.1.5) between CADP, the TINA toolbox developed by SVF, and the VERCORS platform developed by OASIS. We also undertook the definition of the FIACRE intermediate model (see § 5.2.3).

Additionally, we collaborated in 2006 with several INRIA project teams:

- HELIX (Rhône-Alpes): applications of model checking to biological systems (Estelle Dumas, Hidde de Jong, Pedro Monteiro, Michel Page, and Adrien Richard);
- OASIS (Sophia-Antipolis): collaboration in the framework of the FIACRE national action (Antonio Cansado and Eric Madelaine);
- POPART (Rhône-Alpes): combination of the CADP and PROMETHEUS compositional verification tools (Gregor Goessler).

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

- LAAS-CNRS laboratory (Toulouse): collaboration in the framework of the FIACRE, OPENEMBEDD, and TOPCASED projects (Bernard Berthomieu and François Vernadat);
- LIP laboratory (Lyon): until November 2006, R. Mateescu had a part-time (20%) collaboration with the PLUME team;
- LE2I laboratory (Dijon): since December 2006, R. Mateescu is hosted by the computer science team of LE2I.

7.2. International Collaborations

The VASY project team of INRIA and the SEN2 team of CWI collaborate in SENVA, a joint research team on safety-critical systems (see http://www.inrialpes.fr/vasy/senva). Launched in 2004, the SENVA team is supported by INRIA's European and International Affairs Department and by CWI. The first three years of SENVA have been favorably evaluated by a panel of international experts in November 2006.

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

H. Garavel is a member of IFIP (*International Federation for Information Processing*) Technical Committee 1 (*Foundations of Computer Science*) Working Group 1.8 on Concurrency Theory, launched in 2005 and chaired by Luca Aceto.

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

In addition to our partners in aforementioned contractual collaborations, we had scientific relations in 2006 with several international universities and research centers, including:

- Imperial College (Jeff Kramer and Jeff Magee),
- University of Málaga (Carlos Canal and Pedro Merino) [25], [30], and
- University "La Sapienza" of Rome (Antonella Chirichiello and Benjamin Habegger) [26].

7.3. Visits and Invitations

In 2006, we had the following scientific exchanges:

- Jaco van de Pol (CWI, Amsterdam, The Netherlands) visited us on January 26, 2006.
- Christian Attiogbé (University of Nantes) visited us on March 6, 2006 and gave a talk entitled "*Composants logiciels : spécification, composition et vérification avec Kmelia*".
- Pascal Poizat (University of Evry – Val d'Essonne) visited us on May 15–19, 2006. He gave a talk entitled "*Adaptation logicielle – une approche automatisée basée sur des expressions régulières de vecteurs de synchronisation*".

- The annual SENVA seminar was held in Venosc on June 12–14, 2006. In addition to the VASY project team, Manuel Baclet (ENSEEIHT), Jens Calamé, Natalia Ioustinova, Jaco van de Pol, Michael Weber, and Anton Wijs (CWI, Amsterdam), Wan Fokking (Free University of Amsterdam), Rodolfo S. Gomez and Li Su (University of Kent), and Sylvain Peyronnet (EPITA) attended this seminar. The list of talks is available from http://www.inrialpes.fr/vasy/senva/workshop2006.
- Sylvie Lesmanne, Jacques Abily, and Azedine Abdelli (BULL) visited us on June 29–30, 2006.

8. Dissemination

8.1. Software Dissemination and Internet Visibility

The VASY project team distributes two main software tools: the CADP toolbox (see § 4.1) and the TRAIAN compiler (see § 4.2). In 2006, the main facts are the following:

- We prepared and distributed 15 successive beta-versions (2004-h, ..., 2004-k, 2005-a, ..., 2005-k) of CADP, leading to a stable version named CADP 2006 "Edinburgh", released on December 12, 2006.
- The number of license contracts signed for CADP increased from 345 to 366.
- We were requested to grant CADP licenses for 822 different computers in the world.
- The TRAIAN compiler was downloaded by 59 different sites.

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

8.2. Program Committees

In 2006, the members of VASY assumed the following responsibilities:

- H. Garavel was, together with John Hatcliff (Kansas State University), responsible for a special issue [19] of the STTT (*Software Tools for Technology Transfer*) journal, which gathers the best software-oriented papers of TACAS'2003.
- H. Garavel was, together with John Hatcliff (Kansas State University), responsible for a special issue [18] of the TCS (*Theoretical Computer Science*) journal, which gathers the best theory-oriented papers of TACAS'2003.
- H. Garavel was a steering committee member of the PDMC (*Parallel and Distributed Methods in Verification*) series of international workshops.
- H. Garavel was a program committee member of TACAS'2006 (*12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Vienna, Austria, March 25 – April 2, 2006).
- R. Mateescu was a program committee member of MSVVEIS'2006 (*4th International Workshop on Modeling, Simulation, Verification, and Validation of Enterprise Information Systems*, Paphos, Cyprus, May 23–24, 2006).
- R. Mateescu was a program committee member of ICCGI'2006 (*International Conference on Computing in the Global Information Technology*, Bucharest, Romania, August 1–3, 2006).
- R. Mateescu was a program committee member of FMICS'2006 (*11th International Workshop on Formal Methods for Industrial Critical Systems*, Bonn, Germany, August 26–27, 2006).
- H. Garavel was a program committee member of PDMC'2006 (*5th International Workshop on Parallel and Distributed Methods in Verification*, Bonn, Germany, August 31, 2006).
- R. Mateescu was a program committee member of EWSA'2006 (*3rd European Workshop on Software Architectures*, Nantes, France, September 4–5, 2006).

- R. Mateescu was a program committee member of CAL'2006 (*1ère Conférence Francophone sur les Architectures Logicielles*, Nantes, France, September 6–8, 2006).
- H. Garavel was a program committee member of FORTE'2006 (*26th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems*, Paris, France, September 26–29, 2006).
- R. Mateescu was a program committee member of ICSEA'2006 (*International Conference on Software Engineering Advances*, Tahiti, French Polynesia, October 29 – November 1st, 2006).

8.3. Lectures and Invited Conferences

In 2006, we gave talks in several international conferences and workshops (see bibliography below). Additionally:

- R. Mateescu gave a talk entitled "*Vérification à la volée basée sur les systèmes d'équations booléennes*" at the LIFC laboratory (Besançon, France) on January 19, 2006.
- R. Mateescu gave a talk entitled "*Modélisation et analyse des systèmes parallèles asynchrones*" at the LE2I laboratory (Dijon, France) on February 8, 2006.
- G. Salaün visited the research group of professors Jeff Kramer and Jeff Magee, Imperial College (London, UK) between January 19 and February 17, 2006.
- F. Lang gave a talk entitled "*Propositions d'extensions temporisées pour* NTIF" at ENST (Paris, France) on February 13–14, 2006.
- G. Salaün gave a talk entitled "*How Process Algebra Can Contribute to the Formal Development of* WEB *Services*" at the University of Málaga (Spain) on March 16, 2006.
- H. Garavel and R. Mateescu visited CWI (Amsterdam, The Netherlands) on April 2–4, 2006. R. Mateescu gave a talk entitled "*Sequential and Distributed Test Generation using Boolean Equation Systems*". H. Garavel gave a talk entitled "*State Space Reduction for Process Algebra Specifications*" and a position statement on future projects.
- H. Garavel gave a talk entitled "*Validation d'architectures multiprocesseurs à l'aide des outils* CADP" at the LABRI laboratory (Bordeaux, France) on April 27, 2006.
- H. Garavel gave a talk entitled "*Validation d'architectures multi-processeurs : 10 ans de collaboration* BULL-INRIA" at CEA/DAM (Bordeaux, France) on April 28, 2006.
- F. Lang gave two talks entitled "*An Overview of* CADP *2006*" and "*Translating the* LOTOS NT *Data Part into* LOTOS *Abstract Data Types*" at INRIA Sophia-Antipolis (France) on July 6–7, 2006.
- W. Serwe gave a talk entitled "*An Overview of* CADP *2006*" at the Dagstuhl seminar Nr. 06351 on August 27–September 1st, 2006.
- G. Salaün gave a talk entitled "*Translating* CHP *into* LOTOS *for the Verification of Asynchronous Hardware Designs with* CADP" at Microsoft Research (Cambridge, United Kingdom) on November 2, 2006.
- H. Garavel and R. Mateescu visited CWI (Amsterdam, The Netherlands) on November 6–9, 2006. R. Mateescu gave a PAM (*Process Algebra Meeting*) talk entitled "CÆSAR_SOLVE*: A Generic On-the-Fly Solver for Alternation-Free Boolean Equation Systems and its Applications to Verification*" on November 8, 2006.
- H. Garavel gave a talk entitled "*Practical applications of process calculi in industrial projects*" at the LIX Colloquium on Emerging Trends in Concurrency Theory (Ecole Polytechnique, Palaiseau, France) on November 13–15, 2006.
- F. Lang gave a talk entitled "*Refined Interfaces for Compositional Verification*" at LAAS-CNRS laboratory (Toulouse, France) on November 20–21, 2006.
- G. Salaün gave a talk entitled "*Software Adaptation: An Approach based on Synchronization Vectors and Regular Expressions*" at the University of Málaga (Spain) on November 22, 2006.
- H. Garavel gave a talk entitled "CADP *2006 from a Model Driven Perspective*" at INRIA Rennes (France) on November 22–23, 2006.

8.4. Teaching Activities

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

In 2006:

- F. Lang and W. Serwe gave the course on "*Temps Réel*" to the 3rd year students of ENSIMAG (18 hours).
- R. Mateescu was a jury member of Gavril Godza's PhD thesis entitled "*Contribuţii la elaborarea sistemelor distribuite tolerante la defecte*", defended at Polytechnic University of Bucharest (Romania) on February 27, 2006.
- F. Lang was a jury member of Ylies Falcone's MSc thesis (DEA) entitled "*Un cadre formel pour le test de politiques de sécurité*", defended at Université Joseph Fourier (Grenoble) on June 21, 2006.
- F. Lang supervised, jointly with Gregor Goessler (POPART project team), the MSc thesis of A. M. Khan entitled "*Connection of Compositional Verification Tools for Embedded Systems*", defended at Université Joseph Fourier (Grenoble) on June 21, 2006.
- R. Mateescu was a jury member of Rémi Brochenin's MSc thesis entitled "*Techniques d'automates pour raisonner sur la mémoire*", defended at Ecole Normale Supérieure de Lyon on June 26, 2006.
- R. Mateescu supervised the internship (*mémoire d'ingénieur*) of D. Thivolle entitled "*Développement d'un évaluateur pour une logique temporelle étendue*", defended at EPITA (Paris) on July 3, 2006.
- H. Garavel was a jury member of Marie Lalire's PhD thesis [15] entitled "*Développement d'une notation algorithmique pour le calcul quantique*", defended at Institut National Polytechnique de Grenoble on October 19, 2006.
- R. Mateescu was a jury member of Emilie Oudot's PhD thesis entitled "*Contributions à la vérification incrémentale des systèmes temporisés à composants*", defended at Université de Franche Comté (Besançon) on December 7, 2006.
- R. Mateescu was elected suppliant member of the "*Commission de spécialistes*" at Université de Bourgogne (section 27).
- H. Garavel was a jury member of Leila Kloul's habilitation thesis entitled "*From Performance Analysis to Performance Engineering: Some Ideas and Experiments*", defended at Université de Versailles-Saint-Quentin-en-Yvelines on December 8, 2006.
- H. Garavel supervised the internship (*mémoire* CNAM) of J. Fereyre entitled "*Conception et amélioration d'outils logiciels pour la vérification distribuée*", to be defended in 2007.
- F. Lang and H. Garavel supervised the internship (*mémoire* CNAM) of N. Lépy entitled "*Environnement de développement intégré sous Eclipse pour la vérification des systèmes concurrents*", to be defended in 2007.

8.5. Miscellaneous Activities

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

H. Garavel is a member of the computing facilities committee of INRIA Rhône-Alpes.

H. Garavel is a member of the recruitment committees for "*chargés de recherche 2ème classe*" and "*ingénieurs associés*" at INRIA Rhône-Alpes.

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

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

F. Lang leads the working group (5 persons) in charge of proposing a new distribution of offices among the research and administrative teams located in the INRIA building of Montbonnot.

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

9. Bibliography

Major publications by the team in recent years

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

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

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

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

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

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

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

[8] H. GARAVEL, R. MATEESCU, I. SMARANDACHE-STURM. *Parallel State Space Construction for Model-Checking*, in "Proceedings of the 8th International SPIN Workshop on Model Checking of Software SPIN'2001 (Toronto, Canada), Berlin", M. B. DWYER (editor). , Lecture Notes in Computer Science, Full version available as Inria Research Report RR-4341, vol. 2057, Springer Verlag, May 2001, p. 217–234, http://hal.inria.fr/inria-00072247.

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

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

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

[12] H. GARAVEL, C. VIHO, M. ZENDRI. *System Design of a CC-NUMA Multiprocessor Architecture using Formal Specification, Model-Checking, Co-Simulation, and Test Generation*, in "Springer International Journal on Software Tools for Technology Transfer (STTT)", Full version available as Inria Research Report RR-4041, vol. 3, n^{o} 3, July 2001, p. 314–331, http://hal.inria.fr/inria-00072597.

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

[14] G. SALAÜN, W. SERWE. *Translating Hardware Process Algebras into Standard Process Algebras — Illustration with CHP and LOTOS*, in "Proceedings of the 5th International Conference on Integrated Formal Methods IFM'2005 (Eindhoven, The Netherlands)", J. VAN DE POL, J. ROMIJN, G. SMITH (editors). , Lecture Notes in Computer Science, Full version available as Inria Research Report RR-5666, vol. 3771, Springer Verlag, November 2005, p. 287–306.

Year Publications

Doctoral dissertations and Habilitation theses

[15] M. LALIRE. *Développement d'une notation algorithmique pour le calcul quantique*, Thèse de Doctorat, Institut National Polytechnique de Grenoble, October 2006.

Articles in refereed journals and book chapters

[16] C. ATTIOGBÉ, P. POIZAT, G. SALAÜN. *A Formal and Tool-Equipped Approach for the Integration of State Diagrams and Formal Datatypes*, in "IEEE Transactions on Software Engineering", to appear, 2007.

[17] A. CHIRICHIELLO, G. SALAÜN. *Encoding Process Algebraic Descriptions of Web Services into BPEL*, in "International Journal on Web Intelligence and Agent Systems", to appear, 2007.

[18] H. GARAVEL, J. HATCLIFF. *TACAS 2003 Special Issue — Preface*, in "Theoretical Computer Science", vol. 354, n^{o} 2, March 2006, p. 169–172.

[19] H. GARAVEL, J. HATCLIFF. *Why you should definitely read this special section*, in "Springer International Journal on Software Tools for Technology Transfer (STTT)", vol. 8, n^{o} 1, February 2006, p. 1–3.

[20] H. GARAVEL, W. SERWE. *State Space Reduction for Process Algebra Specifications*, in "Theoretical Computer Science", vol. 351, n^{o} 2, February 2006, p. 131–145.

[21] F. LANG. *Explaining the Lazy Krivine Machine Using Explicit Substitution and Addresses*, in "Journal of Higher-Order and Symbolic Computation, special issue on Krivine's machine", to appear, 2007.

[22] R. MATEESCU. *CAESAR_SOLVE: A Generic Library for On-the-Fly Resolution of Alternation-Free Boolean Equation Systems*, in "Springer International Journal on Software Tools for Technology Transfer (STTT)", Full version available as Inria Research Report RR-5948, July 2006, vol. 8, n^{o} 1, February 2006, p. 37–56, https://hal.inria.fr/inria-00084628.

[23] R. MATEESCU. *Modélisation et analyse de systèmes asynchrones avec CADP*, N. NAVET (editor). , Traité IC2, Full version available as Inria Research Report RR 5953, chap. 5, Lavoisier, 2006, p. 151–180, https://hal.inria.fr/inria-00088076.

[24] G. SALAÜN, L. BORDEAUX, M. SCHAERF. *Describing and Reasoning on Web Services using Process Algebra*, in "International Journal of Business Process Integration and Management", vol. 1, n^{o} 2, 2006, p. 116–128.

Publications in Conferences and Workshops

[25] C. CANAL, P. POIZAT, G. SALAÜN. *Synchronizing Behavioural Mismatch in Software Composition*, in "Proceedings of the 8th IFIP International Conference on Formal Methods for Open Object-based Distributed Systems FMOODS'2006 (Bologna, Italy)", R. GORRIERI, H. WEHRHEIM (editors). , Lecture Notes in Computer Science, vol. 4037, Springer Verlag, June 2006, p. 63–77.

[26] A. CHIRICHIELLO, G. SALAÜN. *Formal Development of Web Services*, in "Proceedings of the 4th International Workshop on Artificial Intelligence for Service Composition AISC'06 (Trento, Italy)", August 2006, p. 36–43.

[27] H. GARAVEL, R. MATEESCU, D. BERGAMINI, A. CURIC, N. DESCOUBES, C. JOUBERT, I. SMARANDACHE-STURM, G. STRAGIER. *DISTRIBUTOR and BCG_MERGE: Tools for Distributed Explicit State Space Generation*, in "Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2006 (Vienna, Austria)", H. HERMANNS, J. PALBERG (editors). , Lecture Notes in Computer Science, vol. 3920, Springer Verlag, March–April 2006, p. 445–449.

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

[29] F. LANG. *Refined Interfaces for Compositional Verification*, in "Proceedings of the 26th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2006 (Paris, France)", E. NAJM, J.-F. PRADAT-PEYRE, V. VIGUIÉ DONZEAU-GOUGE (editors). , Lecture Notes in Computer Science, Full version available as Inria Research Report RR-5996, vol. 4229, Springer Verlag, September 2006, p. 159–174.

[30] P. POIZAT, C. CANAL, G. SALAÜN. *Adaptation logicielle: une approche basée sur des expressions régulières de vecteurs de synchronisation*, in "Proceedings of 1ère Conférence francophone sur les Architectures Logicielles CAL'06 (Nantes, France)", M. C. OUSSALAH, F. OQUENDO, D. TAMZALIT, T. KHAMMACI (editors). , Hermes Science, September 2006, p. 31–39.

[31] P. POIZAT, J.-C. ROYER, G. SALAÜN. *Bounded Analysis and Decomposition for Behavioural Descriptions of Components*, in "Proceedings of the 8th IFIP International Conference on Formal Methods for Open Object-based Distributed Systems FMOODS'2006 (Bologna, Italy)", R. GORRIERI, H. WEHRHEIM (editors). , Lecture Notes in Computer Science, vol. 4037, Springer Verlag, June 2006, p. 33–47.

[32] P. POIZAT, G. SALAÜN, M. TIVOLI. *An Adaptation-based Approach to Incrementally Build Component Systems*, in "Proceedings of the 3rd International Workshop on Formal Aspects of Component Software FACS'06 (Prague, Czech Republic)", F. DE BOER, V. MENCL (editors). , Electronic Notes in Theoretical Computer Science, September 2006.

[33] P. POIZAT, G. SALAÜN, M. TIVOLI. *On Dynamic Reconfiguration of Software Adaptations*, in "Proceedings of the 3rd International Workshop on Coordination and Adaptation for Software Entities WCAT'06 (Nantes, France)", July 2006.

[34] G. SALAÜN, W. SERWE, Y. THONNART, P. VIVET. *Formal Verification of CHP Specifications with CADP — Illustration on an Asynchronous Network-on-Chip*, in "Proceedings of the 13th IEEE International Symposium on Asynchronous Circuits and Systems ASYNC 2007 (Berkeley, California, USA)", to appear, IEEE Computer Society Press, 2007.

Miscellaneous

[35] D. CHAMPELOVIER, H. GARAVEL. *Reference Manual of the* LOTOS NT *to* LOTOS *Translator — Version 2E*, Inria/Vasy, 47 pages, September 2006.

[36] A. M. KHAN. *Connection of Compositional Verification Tools for Embedded Systems*, Mémoire master 2 recherche, Université Joseph Fourier, Grenoble, June 2006.

References in notes

[37] B. K. AICHERNIG, C. C. DELGADO. *From Faults via Test Purposes to Test Cases: On the Fault-Based Testing of Concurrent Systems*, in "Proceedings of the 9th International Conference on Fundamental Approaches to Software Engineering FASE'06 (Vienna, Austria)", L. BARESI, R. HECKEL (editors). , Lecture Notes in Computer Science, n^{o} 3922, Springer Verlag, March 2006, p. 324–338.

[38] P. ANDRÉ, G. ARDOUREL, C. ATTIOGBÉ. *Spécification d'architectures logicielles en Kmélia: hiérarchie de connexion et composition*, in "Proceedings of 1ère Conférence Francophone sur les Architectures Logicielles CAL'06 (Nantes, France)", F. OQUENDO, M. OUSSALAH (editors). , Hermès Science/Lavoisier, September 2006.

[39] C. ATTIOGBÉ, G. ARDOUREL, P. ANDRÉ. *Checking Component Composability*, in "Proceedings of the 5th International Symposium on Software Composition SC'06 (Vienna, Austria)", W. LOWE, M. SUDHOLT (editors). , Lecture Notes in Computer Science, n^{o} 4089, Springer Verlag, March 2006, p. 18–33.

[40] T. BARROS, A. CANSADO, E. MADELAINE, M. RIVERA. *Model-Checking Distributed Components: The Vercors Platform*, in "Proceedings of the 3rd International Workshop on Formal Aspects of Component

Software FACS'06 (Prague, Czech Republic)", Electronic Notes in Theoretical Computer Science, Elsevier, September 2006.

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

[42] F. BELLEGARDE, J. JULLIAND, H. MOUNTASSIR, E. OUDOT. *The Tool VeSTA: Verification of Simulations for Timed Automata*, Technical Report, n^{o} RT2006-01, LIFC, Université de Franche-Comté, Besançon, France, July 2006.

[43] C. BENAC EARLE, L.-Å. FREDLUND. *Verification of Language Based Fault-Tolerance*, in "Proceedings of the 10th International Conference on Computer Aided Systems Theory EUROCAST 2005 (Las Palmas de Gran Canaria, Spain)", R. MORENO-DÍAZ, F. PICHLER, A. QUESADA-ARENCIBIA (editors). , Lecture Notes in Computer Science, vol. 3643, Springer Verlag, February 2005, p. 140–149.

[44] C. BENAC EARLE, L.-Å. FREDLUND, J. DERRICK. *Verifying Fault-Tolerant Erlang Programs*, in "Proceedings of the 2005 ACM SIGPLAN Workshop on Erlang (Tallinn, Estonia)", K. F. SAGONAS, J. ARMSTRONG (editors). , ACM Press, September 2005, p. 26–34.

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

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

[47] H. BOWMAN, R. GOMEZ. *Concurrency Theory: Calculi and Automata for Modelling Untimed and Timed Concurrent Systems*, Springer Verlag, 2006.

[48] E. BÖDE, M. HERBSTRITT, H. HERMANNS, S. JOHR, T. PEIKENKAMP, R. PULUNGAN, R. WIMMER, B. BECKER. *Compositional Performability Evaluation for Statemate*, in "Proceedings of the 3rd International Conference on the Quantitative Evaluation of Systems QUEST'06 (Riverside, California, USA)", IEEE Computer Society Press, September 2006, p. 167–178.

[49] A. CALIXTO, R. MONROY. *Formal Analysis of TLS*, in "Studia Informatica Universalis", vol. 2, n^{o} 2, 2002, p. 235–249.

[50] G. CHEHAIBAR, H. GARAVEL, L. MOUNIER, N. TAWBI, F. ZULIAN. *Specification and Verification of the PowerScale Bus Arbitration Protocol: An Industrial Experiment with LOTOS*, in "Proceedings of the Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, and Protocol Specification, Testing, and Verification FORTE/PSTV'96 (Kaiserslautern, Germany)", R. GOTZHEIN, J. BREDEREKE (editors). , Full version available as Inria Research Report RR-2958, Chapman & Hall, IFIP, October 1996, p. 435–450, http://hal.inria.fr/inria-00073740.

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

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

[53] C. C. DELGADO, B. K. AICHERNIG. *Test Purpose Generation by Specification Mutation in Distributed Systems*, Technical report, n^{o} 313, International Institute for Software Technology, United Nations University, Macau (China), September 2004.

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

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

[56] M. HUISMAN, P. WORAH, K. SUNESEN. *A Temporal Logic Characterisation of Observational Determinism*, in "Proceedings of the 19th IEEE Computer Security Foundations Workshop CSFW'06 (Venice, Italy)", IEEE Computer Society Press, July 2006.

[57] IEEE. *IEEE Standard SystemC Language Reference Manual*, IEEE Standard, n^{o} 1666-2005, Institution of Electrical and Electronic Engineers, December 2005.

[58] H. JONKER, S. K. NAIR, M. T. DASHTI. *Nuovo DRM Paradiso: Formal Specification and Verification of a DRM Protocol*, in "Proceedings of the 1st Benelux Workshop on Information and System Security WISSec 2006 (Antwerpen, Belgium)", Lecture Notes in Computer Science, Springer Verlag, November 2006.

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

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

[61] M. MOY, F. MARANINCHI, L. MAILLET-CONTOZ. PINAPA*: An Extraction Tool for SystemC descriptions of Systems-on-a-Chip*, in "EMSOFT", W. WOLF (editor). , ACM, September 2005.

[62] E. OUDOT. *Contributions à la vérification incrémentale des systèmes temporisés à composants*, Ph. D. Thesis, Université de Franche-Comté, December 2006.

[63] B. PLOEGER, L. SOMERS. *Analysis and Verification of an Automatic Document Feeder*, CS-Report, n^{o} 06–25, Eindhoven University of Technology, 2006.

[64] O. PONSINI, C. FÉDÈLE, E. KOUNALIS. *Rewriting of imperative programs into logical equations*, in "Science of Computer Programming", vol. 56, n^{o} 3, May – June 2005, p. 363–401.

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

[66] J. J. SÁNCHEZ PENAS. *From Software Architecture to Formal Verification of a Distributed System*, Ph. D. Thesis, University of Coruña (Spain), November 2006.

[67] A. J. WIJS, J. VAN DE POL. *Solving Scheduling Problems by Untimed Model Checking — The Clinical Chemical Analyser Case Study*, in "Proceedings of the 10th International Workshop on Formal Methods for Industrial Critical Systems FMICS'05 (Lisbon, Portugal)", Also available as CWI Technical Report SEN-R0608, ACM, September 2005, p. 54–61.

[68] A. J. WIJS, J. VAN DE POL. *Solving Scheduling Problems by Untimed Model Checking*, Technical Report, n^{o} SEN-R0608, CWI, Amsterdam, The Netherlands, 2006.

[69] M. DEL MAR GALLARDO, C. JOUBERT, P. MERINO. *Implementing Influence Analysis using Parameterised Boolean Equation Systems*, in "Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation ISOLA'06 (Paphos, Cyprus)", IEEE Computer Society Press, November 2006.

[70] M. DEL MAR GALLARDO, P. MERINO, D. SANÁN. *Towards Model Checking C Code with OPEN/CÆSAR*, in "Proceedings of the 4th International Workshop on Modelling, Simulation, Verification, and Validation of Enterprise Information Systems MSVVEIS'06 (Paphos, Cyprus)", J. BARJIS, U. ULTES-NITSCHE, J. C. AUGUSTO (editors). , INSTICC Press, May 2006, p. 198–201.