INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team VASY

Validation of Systems

Rhône-Alpes

THEME COM

Contents

1 Team

Head of team

Hubert Garavel [DR2 INRIA]

Administrative Assistants

Valérie Gardès [since February 2, 2004]

Catherine Magnin [until January 30, 2004]

Inria Staff

Radu Mateescu [CR1 INRIA]

Frédéric Lang [CR1 INRIA]

Wendelin Serwe [CR2 INRIA, since October 1st, 2004]

Bull Staff

Solofo Ramangalahy [BULL engineer, until May 31, 2004]

Software Engineers

Damien Bergamini

David Champelovier

Nicolas Descoubes [until November 19, 2004]

Post-Doctoral Fellows

Aurore Collomb [until June 30, 2004]

Gwen Salaün [since October 1st, 2004]

Wendelin Serwe [until September 30, 2004]

Ph. D. Student

Christophe Joubert

2 Overall Objectives

2.1 Introduction

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

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

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

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

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

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

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

2.2 Models and Verification Techniques

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

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

- *Behavioral properties* express the intended functioning of the system in the form of automata (or higher level descriptions, which are then translated into automata). In such

a case, the natural approach to verification is *equivalence checking*, which consists in comparing the system model and its properties (both represented as automata) modulo some equivalence or preorder relation. We develop equivalence checking tools that compare and minimize automata modulo various equivalence and preorder relations; some of these tools also apply to stochastic and probabilistic models (such as Markov chains).

- *Logical properties* express the intended functioning of the system in the form of temporal logic formulas. In such a case, the natural approach to verification is *model checking*, which consists in deciding whether the system model satisfies or not the logical properties. We develop model checking tools for a powerful form of temporal logic, the *modal μ-calculus*, which we extend with typed variables and expressions so as to express predicates over the data contained in the model. This extension (the practical usefulness of which was highlighted in many examples) provides for properties that could not be expressed in the standard μ-calculus (for instance, the fact that the value of a given variable is always increasing along any execution path).

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

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

2.3 Languages and Compilation Techniques

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

We are working on several languages with the above properties:

- LOTOS is an international standard for protocol description (ISO/IEC standard 8807:1989), which combines the concepts of process algebras (in particular CCS and CSP) and algebraic abstract data types. Thus, LOTOS can describe both asynchronous concurrent processes and complex data structures.

  We use LOTOS for various industrial case studies and we develop LOTOS compilers, which are part of the CADP toolbox (see § 4.1).

- Between 1992 and 2001, we contributed to the revision of LOTOS undertaken within ISO. This led to the definition of E-LOTOS (*Enhanced*-LOTOS, ISO/IEC standard 15437:2001),

which tries to provide a greater expressiveness (for instance, by introducing quantitative time to describe systems with real-time constraints) together with a better user friendliness.

Our contributions to E-LOTOS are available on the WEB (see `http://www.inrialpes.fr/vasy/elotos`).

- We are also working on an E-LOTOS variant, named LOTOS NT (LOTOS *New Technology*) [7, 12], in which we can experiment new ideas more freely than in the constrained framework of an international standard. Like E-LOTOS, LOTOS NT consists of three parts: A *data part*, which allows the description of data types and functions, a *process part*, which extends the LOTOS process algebra with new constructs such as exceptions and quantitative time, and *modules*, which provide for structure and genericity. Both languages differ in that LOTOS NT combines imperative and functional features, and is also simpler than E-LOTOS in some respects (static typing, operator overloading, arrays), which should make it easier to implement.

  We are developing for LOTOS NT a prototype compiler named TRAIAN (see § 4.2).

2.4 Implementation and Experimentation

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

3 Application Domains

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

- *Hardware architectures:* asynchronous circuits, bus arbitration protocols, cache coherency protocols, hardware/software codesign;
- *Databases:* transaction protocols, distributed knowledge bases, stock management;
- *Consumer electronics:* audiovisual remote control, video on-demand, FIREWIRE bus, home networking;
- *Security protocols:* authentication, electronic transactions, cryptographic key distribution;
- *Embedded systems:* smart-card applications, air traffic control;
- *Distributed systems:* virtual shared memory, distributed file systems, election algorithms, dynamic reconfiguration algorithms, fault tolerance algorithms;

- *Telecommunications:* high speed networks, network management, mobile telephony, feature interaction detection;
- *Human-machine interaction:* graphical interfaces, biomedical data visualization, etc.

4 Software

4.1 The CADP Toolbox

Participants: Damien Bergamini, David Champelovier, Nicolas Descoubes, 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 [10] 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 [6] 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 [11] is a generic software environment for developing tools that explore graphs on the fly (for instance, simulation, verification, and test generation tools). Such tools can be developed independently from any particular high level language. In this respect, OPEN/CÆSAR plays a central role in CADP by connecting language-oriented tools with model-oriented tools. OPEN/CÆSAR provides a set of libraries with their programming interfaces, as well as various tools, such as:
  - BISIMULATOR [19], which checks bisimulation equivalences and preorders on the fly,
  - DETERMINATOR, which eliminates nondeterminism in normal, probabilistic, or stochastic systems,
  - EVALUATOR [13], which evaluates regular alternation-free μ-calculus formulas,
  - EXECUTOR, which performs random execution,
  - EXHIBITOR, which searches for execution sequences matching a given regular expression,
  - GENERATOR and REDUCTOR, which construct the graph of reachable states,
  - PROJECTOR, which computes abstractions of communicating systems,

  - 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_MIN, which minimizes a graph modulo strong or branching equivalences (and can also deal with probabilistic and stochastic systems),
  - BCG_STEADY, which performs steady-state numerical analysis of (extended) continuous-time Markov chains,
  - BCG_TRANSIENT, which performs transient numerical analysis of (extended) continuous-time Markov chains, and
  - XTL (*eXecutable Temporal Language*), which is a high level, functional language for programming exploration algorithms on BCG graphs. XTL provides primitives to handle states, transitions, labels, *successor* and *predecessor* functions, etc. For instance, one can define recursive functions on sets of states, which allow to specify in XTL evaluation and diagnostic generation fixed point algorithms for usual temporal logics (such as HML [HM85], CTL [CES86], ACTL [NV90], 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.

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

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

[NV90] R. D. NICOLA, F. W. VAANDRAGER, *Action versus State Based Logics for Transition Systems*, *Lecture Notes in Computer Science*, *469*, Springer Verlag, 1990, p. 407–419.

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

The CADP tools are well-integrated and can be accessed easily using either the EUCALYPTUS graphical interface or the SVL [3] 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 found useful applications in compiler construction [2]. The recent compilers developed by the VASY team — namely AAL (see § 5.1.4), EVALUATOR 4.0, EXP.OPEN 2.0 (see § 5.1.5), NTIF (see § 5.2.3), 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 OPEN/CÆSAR Libraries

Participants: Nicolas Descoubes, Hubert Garavel, Wendelin Serwe.

The OPEN/CÆSAR libraries [11] are useful modules for on the fly verification, such as state tables, stacks, bitmap tables, etc. These libraries play a discrete, yet central role in the CADP toolbox.

In 2004, we improved the OPEN/CÆSAR libraries in various ways:

- A new library named CAESAR_AREA was developed (800 lines of C code), which provides for genericity by allowing different objects (states, labels, character strings, user-defined memory blocks) to be handled uniformly.
- Another new library named CAESAR_MASK was developed (1,260 lines of C code), which exports primitives for applying sequences of hiding and renaming operations (defined using regular expressions) to memory blocks. In particular, this library allows to hide and/or rename labels on the fly. The DETERMINATOR, GENERATOR, PROJECTOR, and REDUCTOR tools have been enhanced with this new functionality, and the code of the BCG_LABELS tool was greatly simplified by using the new CAESAR_MASK library.
- The existing library CAESAR_TABLE was enhanced by taking advantage of the generic features of CAESAR_AREA, which resulted in simpler and more expressive programming interfaces. Also, a delicate problem regarding automatic insertion of padding bytes to satisfy alignment constraints was solved. In spite of all this changes, binary compatibility was preserved so as not to impact existing OPEN/CÆSAR tools developed by third-parties.

5.1.2 The CÆSAR_SOLVE Library

Participant: Radu Mateescu.

CÆSAR_SOLVE is a generic software library for solving boolean equation systems of alternation depth 1 (i.e., without mutual recursion between minimal and maximal fixed point equations) on the fly. This library is at the core of several CADP verification tools, namely the equivalence checker BISIMULATOR (see § 5.1.3), the model checker EVALUATOR 4.0, and the τ-confluence reduction tool. The resolution method is based on boolean graphs, which provide an intuitive representation of dependencies between boolean variables; boolean graphs are handled implicitly in a way similar to the OPEN/CÆSAR interface.

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

In 2004, the CÆSAR_SOLVE library (9,640 lines of C code) was extended and improved as follows:

- The library was enhanced with a primitive that enumerates all the variables of a boolean equation system that were stabilized (i.e., whose final value was determined) during the resolutions performed on the system, either since the last call of the primitive, or since the creation of the system (in the case of the first call of the primitive). This primitive provides useful diagnostic information, e.g., for vacuity detection in model checking, which consists in determining, for a temporal logic formula, its subformulas having the same truth value on all states.

- Several primitives of the library were simplified in order to enhance the user-friendliness of the interface (e.g., the double pointer indirection was eliminated from the arguments of the primitive for creating a boolean equation system) and to take advantage of the CÆSAR_AREA library newly added to OPEN/CÆSAR (see § 5.1.1). Also, error management was improved in order to be compatible with the other libraries of OPEN/CÆSAR.

- The textual format of boolean equation systems was enhanced to allow the specification of all the parameters necessary to the creation and resolution of a system (e.g., resolution algorithm, single or multiple resolutions, etc.). Also, the syntax of variable identifiers was simplified, and the size of the variables of a boolean equation system read from a text file was reduced.

- The CÆSAR_SOLVE library became part of CADP in December 2004. The documentation of the library interface (23 pages) was updated and extended with examples.

A paper about the CÆSAR_SOLVE library was accepted for publication [17].

5.1.3 The BISIMULATOR Tool

Participants: Damien Bergamini, Nicolas Descoubes, Radu Mateescu.

BISIMULATOR is an equivalence checker, which takes as input two graphs to be compared (one represented implicitly using the OPEN/CÆSAR environment, the other represented explicitly as a BCG file) and determines whether they are equivalent (modulo a given equivalence relation) or whether one of them is included in the other (modulo a given preorder relation).

BISIMULATOR works on the fly, meaning that only those parts of the implicit graph pertinent to verification are explored. Thanks to the use of OPEN/CÆSAR, BISIMULATOR can be applied directly to descriptions written in high level languages (for instance, LOTOS). This is a significant improvement compared to older tools (such as ALDÉBARAN and FC2IMPLICIT) which only accept lower level models (networks of communicating automata).

BISIMULATOR works by reformulating the graph comparison problem in terms of a boolean equation system, which is solved using the CÆSAR_SOLVE library (see § 5.1.2). A useful functionality of BISIMULATOR is the generation of diagnostics (counterexamples), which explain why two graphs are not equivalent (or not included one in the other). The counterexamples generated by BISIMULATOR are directed acyclic graphs and usually much smaller than those generated by other tools (such as ALDÉBARAN) that can only generate counterexamples restricted to sets of traces.

In 2004, we continued the development of the BISIMULATOR tool (11,700 lines of C code):

- The tool was enhanced with comparisons modulo the branching equivalence relation and its associated preorder relation. The generation of counterexamples for branching equivalence and its preorder was also implemented.

- An optimized encoding of the branching and observational equivalence relations in terms of boolean equation systems was designed, which exploits local absence of τ-transitions to simplify the right-hand sides of boolean equations. This optimization increased the

performance of BISIMULATOR roughly by a factor 2, both in terms of speed and memory consumption. It is worth noticing that boolean equations can be simplified on the fly: There is no need to check before verification if the explicit graph (represented in BCG format) does not contain τ-transitions.

- The tool was enhanced with two new functionalities, implemented using the primitives newly added to the CÆSAR_SOLVE library (see § 5.1.2): printing of the boolean equation system corresponding to the equivalence checking problem in textual form, and printing of statistical information about its resolution.
- In order to validate BISIMULATOR and compare its performance with other equivalence checking tools, we continued the extensive experiments on realistic examples obtained from the CADP distribution and the VLTS benchmark suite. The systematic use of BISIMULATOR allowed to detect an error in the on the fly algorithm for branching equivalence implemented in the ALDÉBARAN tool.
- The encoding of boolean variables has been adapted in order to connect BISIMULATOR to the new distributed resolution algorithm for boolean equation systems (see § 5.1.6). The new encoding ensures that the contents of the boolean variables, which are sent over the communication network during the distributed resolution, are machine-independent, such that they can be interpreted unambiguously by all the processes involved in the distributed computation.
- The tool was properly documented [33] and integrated in CADP in December 2004.

The BISIMULATOR tool was subject to an accepted publication [19].

5.1.4 The AAL Tool

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

In the framework of the ARCHWARE project (see § 6.1), we focus on the analysis of software architectures.

AAL (*Architecture Analysis Language*) is the language defined by ARCHWARE for expressing properties of software architectures and architectural styles. AAL contains operators borrowed from first-order logic and modal μ-calculus, extended with predicates specific to architectural descriptions. It allows to specify both style-related structural properties (e.g., connectivity between components, cardinality, etc.) and architecture-related behavioral properties (e.g., safety, liveness, fairness).

AAF-MC (*Architecture Analysis Formalism for Model Checking*) is the fragment of AAL containing properties to be verified using model checking. A large number of property patterns relevant to software architectures are available as libraries defined in AAF-MC, and several fragments of AAF-MC, compatible with usual equivalence relations (e.g., strong, branching, observational, and safety equivalences) are identified.

AAF-MC is equipped with a model checker that translates the temporal formulas expressed in AAF-MC into boolean equation systems. The analysis methodology adopted by ARCHWARE

consists in using the AAF-MC model checker to verify correctness properties on execution traces generated during the simulation of an architectural description [20].

In 2004, we continued the development of the AAF-MC model checker (16,400 lines of code):

- The model checker was enhanced with generation of diagnostics (examples and counterexamples) as prefixes of the execution traces on which AAF-MC formulas are verified. This facilitates the understanding of AAF-MC formulas and helps the user to debug an architecture.
- A new verification algorithm was added to the model checker, based upon a breadth-first search resolution strategy of the underlying boolean equation system. Compared to the existing depth-first search based algorithm, the new algorithm has the advantage of producing diagnostics of shortest length.
- In collaboration with ARCHWARE partners, the AAF-MC model checker was integrated in the ARCHWARE environment, which is centered around a virtual machine responsible for executing architectural descriptions written in the ARCHWARE ADL (*Architectural Description Language*). The various ARCHWARE tools (e.g., visual modeler, animator, model checker, etc.) are viewed as COTS (*components off the shelf*) communicating with the virtual machine using WEB services. The AAF-MC model checker was integrated in this environment by developing two software tools:
  - a connector (350 lines of ADL code), which executes on the virtual machine and is responsible for initiating the WEB service connection, launching the model checker on a (possibly) remote machine, and sending the appropriate arguments (execution traces and AAF-MC formulas) encoded as text files;
  - a wrapper (520 lines of JAVA and UNIX shell code), which executes on the machine hosting the model checker and is responsible for receiving the execution traces and the AAF-MC formulas sent by the ADL connector, invoking the AAF-MC model checker, and sending the results (truth value and diagnostic) back to the connector.

The AAF-MC model checker is described in two ARCHWARE deliverables [27, 28]. R. Mateescu gave a keynote presentation about this model checker at VVEIS'2004 (see § 8.3).

5.1.5 Compositional Verification Tools

Participant: 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 is a tool (totally rewritten in 2002) that implements behaviour abstraction [GSL96,KM97], by taking into account interface constraints. In 2004, we improved

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

[KM97] J.-P. KRIMM, L. MOUNIER, "Compositional State Space Generation from LOTOS Programs", *in: Proceedings of TACAS'97 Tools and Algorithms for the Construction and Analysis of Systems*

PROJECTOR 2.0 by adding options to hide and rename labels on the fly, based on the CÆSAR_MASK library, and we corrected a few bugs. A manual page was written for PROJECTOR 2.0 [37] and the tool became part of CADP in December 2004.

EXP.OPEN 2.0 is a tool that explores on the fly the graph corresponding to a network of communicating automata (represented as a set of BCG files). These automata are composed together in parallel using either algebraic operators (as in CCS, CSP, LOTOS, and μCRL), "graphical" operators (as in E-LOTOS [ISO01] and LOTOS NT), or synchronization vectors (as in the MEC and FC2 tools). Additional operators are available to hide and/or rename labels (using regular expressions) and to cut certain transitions. In 2004, we enhanced EXP.OPEN 2.0 along the following lines:

- We took μCRL syntax conventions into account to extract information (gate, offers) from labels, thus improving μCRL support.
- Based on feedback received from external users, we added warning messages when some synchronization between automata cannot ever happen because one of the automata does not contain the appropriate label.
- We implemented a partial order reduction technique for stochastic models (based on observations made in [Her02]), which consists in giving priority to invisible transitions over stochastic transitions, thus expressing that invisible transitions are instantaneous. In collaboration with Holger Hermanns and Sven Johr (Saarland University), we applied this technique to study a stochastic model of a distributed mutual exclusion algorithm (see § 5.3). This allowed to divide by up to 5 the number of states of the generated stochastic models.
- We also implemented a technique that allows to synthesize interface constraints imposed on one automaton by (a subset of) its neighbour automata in a network of communicating automata. These interface constraints can be given to the PROJECTOR 2.0 tool so as to generate the behaviour corresponding to a process. We experimented this technique on two case studies, namely the HAVI (Home Audio Video) protocol developed by eight consumer electronics companies (Grundig, Hitachi, Matsushita, Philips, Sharp, Sony, Thomson, and Toshiba) and a cache coherence protocol (see § 5.3), both modeled in LOTOS. The experiments allowed to reduce the state space of some processes by one or two orders of magnitude, thus improving over existing techniques.
- A detailed manual page for EXP.OPEN 2.0 was written [36] and the tool became part of CADP in August 2004. It is used in the framework of the FIACRE national action (see § 7.1) and at Saarland University, among other places.

In 2004, we enhanced the SVL language and compiler along the following lines:

(University of Twente, Enschede, The Netherlands), E. Brinksma (editor), *Lecture Notes in Computer Science*, *1217*, Springer Verlag, Berlin, April 1997. Extended version with proofs available as Research Report VERIMAG RR97-01.

[ISO01] ISO/IEC, "Enhancements to LOTOS (E-LOTOS)", *International Standard number 15437:2001*, International Organization for Standardization — Information Technology, Genève, September 2001.

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

- SVL now uses the new BISIMULATOR tool (see § 5.1.3) to perform behavioural comparisons;
- SVL now uses PROJECTOR 2.0 to implement its abstraction operator, which allowed to extend this operator with new functionalities;
- SVL now uses EXP.OPEN 2.0 to compute automata products, which allows to perform general label hiding and renaming (previously, only a restricted form of label hiding was available).

5.1.6 Parallel and Distributed Verification Tools

Participants: Damien Bergamini, Nicolas Descoubes, Hubert Garavel, Christophe Joubert, Radu Mateescu.

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

Our initial efforts focused on parallelizing the graph construction algorithm [5], which is a bottleneck for verification as it requires a considerable amount of memory to store all reachable states. In this respect, we developed the following software:

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

In 2004, we improved the distributed model checking tools as follows:

- D. Bergamini fixed a few problems in the CÆSAR_NETWORK library and ported it to the MAC OS X operating system.
- C. Joubert improved the algorithm used in DISTRIBUTOR 3.0 in order to optimize speed by reducing busy waits and to solve an issue in termination detection. N. Descoubes adapted the code of DISTRIBUTOR 3.0 to implement the modified algorithm.

- In the framework of the SENVA collaboration (see § 7.2), Stefan Blom (CWI, Amsterdam), N. Descoubes, and H. Garavel defined the PBG (*Partitioned* BCG *Graphs*) format to represent the concept of "partitioned labeled transition system" advocated in [5]. Then, N. Descoubes and D. Bergamini implemented this new format in DISTRIBUTOR and BCG_MERGE, which simplified the command-line interface of these tools.

- The DISTRIBUTOR and BCG_MERGE tools were ported to the WINDOWS and MAC OS X operating systems, and the manual page of BCG_MERGE was finalized [30].

- C. Joubert continued the development of a distributed version of the CÆSAR_SOLVE library (see § 5.1.2), which uses several machines to solve boolean equation systems on the fly. This distributed version (currently 10,000 lines of C code) implements a resolution algorithm based on a breadth-first traversal of the boolean graph.

  In 2004, this distributed algorithm was improved in order to produce diagnostics (boolean subgraphs illustrating the truth value of boolean variables) and various statistical information about the distributed resolution (e.g., number of variables and dependencies explored, number and size of messages used for resolution and termination detection, computation and idle times for each machine, etc.).

- C. Joubert developed a random generator of boolean equation systems in order to test the performance of the distributed resolution algorithm. This tool (1,000 lines of C code) produces boolean equation systems (represented by the successor function of their corresponding boolean graph) according to various parameters which vary randomly in a given domain: number of variables and operators, number of true and false constants, proportion of disjunctive and conjunctive variables in the right-hand sides of the equations, etc. An extensive set of distributed resolutions of boolean equation systems with various forms produced by the random generator was performed on several clusters of PCs (I-CLUSTER of the APACHE team, IDPOT of the ID-IMAG laboratory, and ION of the SARDES team). These experiments showed a good behaviour of the distributed resolution algorithm: quasi-linear speedup compared to the sequential breadth-first search algorithm of CÆSAR_SOLVE, good scalability with the number of machines, low percentage of termination detection messages, and low memory overhead. For example, a boolean equation system with 240 millions of variables and 1 billion operators was solved in 28 minutes on a cluster with 17 machines.

- C. Joubert and R. Mateescu developed a prototype connection of the BISIMULATOR tool (see § 5.1.3) to the distributed boolean resolution algorithm in order to obtain distributed on the fly equivalence checking functionalities. A large number of experiments were performed on the three clusters of PCs mentioned above, using various graphs obtained from the CADP distribution and the VLTS benchmark suite. Each experiment consisted in comparing a graph with its minimized version modulo a given equivalence relation, which is the worst case for on the fly equivalence checking algorithms, because the two graphs must be explored entirely. For all equivalence relations implemented by BISIMULATOR, the experiments showed speedups close to linear (compared to the sequential version of BISIMULATOR used in breadth-first search mode) and a balanced distribution of work among machines.

Three papers on distributed model checking were either published or accepted for publication [23, 24, 19].

5.1.7 Performance Evaluation Tools

Participants: Damien Bergamini, David Champelovier, Hubert Garavel, Christophe Joubert, Frédéric Lang, Radu Mateescu.

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

In 2004, these tools have progressed as follows:

- D. Bergamini (with some help from H. Garavel) found a subtle semantic bug in the algorithm used by BCG_MIN to minimize a probabilistic/stochastic model with respect to stochastic branching bisimulation.

- D. Champelovier and H. Garavel finalized the BCG_STEADY and BCG_TRANSIENT prototype tools developed in 2002 by C. Joubert and Holger Hermanns (Saarland University). The source code of these tools (2,700 lines of C code) was entirely scrutinized and revised; the command-line interface was enhanced and the various output formats were improved. The draft manual pages were rewritten [31, 32]. These tools became part of CADP in November 2004.

- F. Lang revised the DETERMINATOR prototype tool developed in 2002 by C. Joubert and Holger Hermanns (Saarland University). The source code was revised and two bugs were fixed. R. Mateescu extended DETERMINATOR to eliminate nondeterminism from standard (i.e., non-stochastic) labeled transition systems. The draft manual page was finalized [35] and DETERMINATOR (1,650 lines of C code) became part of CADP in December 2004.

5.1.8 Other Tool Developments

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

We also improved the following CADP tools and libraries:

- D. Champelovier, H. Garavel, and R. Mateescu enhanced the formula language of the EVALUATOR 3.0 model checker by adding concatenation operators for strings and regular expressions. These operators provide a higher degree of parameterization for macro-definitions, which allows to simplify the set of μ-calculus formulas to be verified by

EVALUATOR 3.0. Using these operators, for instance, the set of formulas characterizing a dynamic reconfiguration protocol [CGMdP01] was reduced from 783 to 183 lines.

- W. Serwe reviewed the source code of the TGV tool [JM99] and fixed a bug occurring on WINDOWS platforms only; he also ported TGV to the MAC OS X operating system and removed all compile-time warnings.

- N. Descoubes and H. Garavel solved various issues in the INSTALLATOR tool and enhanced it with new functionalities such as the support for remote connections based on SSH/SCP or KRSH/KCP, and the sending of reminder messages to users before license expiration.

We pursued our continous work of adapting CADP to the latest computing platforms:

- We completed the port undertaken in 2003 of the CADP tools to the latest versions of the LINUX kernel, GNU C library, and GCC 3 compiler.

- With the help of Stefan Blom (CWI, Amsterdam), we ported the CADP tools to AMD's OPTERON processors running in 32-bit mode.

- We improved the integration of the CADP tools with WINDOWS, especially as regards support for file names containing spaces and other blank characters.

- We ported all the CADP tools to the MAC OS X operating system.

We enhanced our software engineering environment used to develop and maintain CADP:

- We merged various lists of pending issues into one unique database (bug repository).

- We set up an automated non-regression testing for all the CADP demo examples and all computing platforms. This revealed a few bugs specific to certain platforms; these bugs, undetected so far, were fixed.

- Additionally, non-regression test suites dedicated to particular tools (namely, BCG_MIN, BCG_STEADY, BCG_TRANSIENT, and DETERMINATOR) were developed.

[CGMdP01] M. A. CORNEJO, H. GARAVEL, R. MATEESCU, N. DE PALMA, "Specification and Verification of a Dynamic Reconfiguration Protocol for Agent-Based Applications", *in: Proceedings of the 3rd IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems DAIS'2001 (Krakow, Poland)*, A. Laurentowski, J. Kosinski, Z. Mossurska, R. Ruchala (editors), IFIP, Kluwer Academic Publishers, p. 229–242, September 2001. Full version available as INRIA Research Report RR-4222.

[JM99] T. JÉRON, P. MOREL, "Test generation derived from model-checking", *in: Proceedings of the Conference on Computer-Aided Verification CAV'99 (Trento, Italy)*, N. Halbwachs, D. Peled (editors), *Lecture Notes in Computer Science*, *1633*, Springer Verlag, p. 108–122, July 1999.

5.2 Languages and Compilation Techniques

5.2.1 Compilation of the LOTOS Data Part

Participants: David Champelovier, Hubert Garavel.

In 2004, we deeply modified the CÆSAR.ADT compiler for the data part of LOTOS so as to implement a feature requested by many users around the world.

Enumerative verification for specifications that contain typed values may require to enumerate exhaustively the domains of certain types, i.e., the set of all values of certain types. To do this, CÆSAR.ADT generates automatically *iterators* (i.e., fragments of C code) suitable for enumerating LOTOS types, which are basically data types defined by a set of free constructors. Obviously, this can only be done for types with a finite domain. CÆSAR.ADT also accepts iterators written by the user manually.

So far, the iterators generated automatically by CÆSAR.ADT were restricted to certain classes of LOTOS types: (bounded) natural numbers, enumerated types, and tuple types. The changes brought to CÆSAR.ADT in 2004 allow iterators to be generated for all finite LOTOS types, including the intricate case of union types, which might be nested at any depth.

This improvement required a deep modification of the concept of iterators and the addition of 1,000 new lines of code in CÆSAR.ADT. In the former version of CÆSAR.ADT, an old-style iterator consisted of one single C macro-definition (similar to a "for"-loop), whereas in the modified versions of CÆSAR.ADT, a new-style iterator consists of two companion C macro-definitions (based on a "first/next" scheme).

A key issue was to maintain, as much as possible, backward compatibility with former versions of CÆSAR.ADT, in particular by accepting old-style iterators already written by the users manually. In most cases, the change is transparent to the end-user; otherwise, error messages are emitted, which will disappear after minor modifications by the user.

The CÆSAR compiler, the predefined LOTOS data type libraries, and the CADP demo examples have been updated so as to take advantage of new-style iterators.

5.2.2 Compilation of the LOTOS Process Part

Participants: Damien Bergamini, Hubert Garavel, Wendelin Serwe.

In 2004, we enhanced in several ways the CÆSAR compiler for the process part of LOTOS:

- We enhanced the optimization by which CÆSAR eliminates all "dead" transitions from its internal network model (i.e., an extended Petri net model generated by CÆSAR from a source LOTOS description).

  Previously, the detection of "dead" transitions was done using an *explicit state* approach, by enumerating all reachable markings. However, a benchmark experiment that we conducted in 2003 on a large LOTOS example provided to us by BULL demonstrated that symbolic methods were superior for that task.

  For this reason, D. Bergamini developed in 2004 a new tool named CÆSAR.BDD [33] (900 lines of C code), which uses symbolic methods (Binary Decision Diagrams) to compute

structural properties of basic Petri nets, including the set of "dead" transitions. The CÆSAR compiler was enhanced to cooperate with CÆSAR.BDD for the elimination of "dead" transitions. The good performance provided by the symbolic approach made possible to turn "dead" transitions elimination into a systematic optimization, whereas it was previously only an optional one.

- We continued the work undertaken in 2003 about techniques for state space reduction, our goal being to decrease the size of the graphs generated by CÆSAR, still preserving strong bisimulation between the original and reduced graphs.

  Our approach is based on live variable analysis, first proposed by H. Garavel and Juan Galvez [Gal93]. The basic idea is to assign a canonical value to any variable that is no longer used, so as to avoid distinguishing state vectors that only differ by the values of some variables not used in the future. This is done by adapting classical data flow analysis to the extended Petri nets generated by CÆSAR and by resetting to zero each variable as soon as it ceases to be alive.

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

  In 2004, we identified a difficult problem arising in the particular case of "*reset/use*" conflicts, and we refined our approach to handle such conflicts properly. This work led to publications [21, 15].

  We implemented our ideas in a prototype version of CÆSAR (about 5,000 lines of additional C code), which we applied to a benchmark suite of 518 LOTOS specifications, among which 289 appeared to be relevant for assessing our approach; for the 229 others, the network variables could be eliminated by optimizations already implemented in CÆSAR, such as constant detection and transformation into registers (i.e., variables local to a transition). For 131 examples, the size of graphs generated by CÆSAR was divided by a mean factor of 9 (with a maximum of 400) as regards the number of states and a mean factor of 13 (with a maximum of 500) as regards the number of transitions. On three further examples our prototype was capable to generate state space that the standard version CÆSAR 6.2 could not handle due to lack of memory. For one of these examples, we observed a reduction factor greater than 10^4.

  Additionally, W. Serwe experimented further uses of data-flow analysis so as to reduce memory requirements for enumerative verification.

5.2.3 Compilation of E-LOTOS

Participants: Aurore Collomb, David Champelovier, Hubert Garavel, Frédéric Lang.

As regards the data part of E-LOTOS, we continued to improve the TRAIAN compiler (see § 4.2), which is distributed on the Internet (see § 8.1) and used intensively within the

[Gal93] J. GALVEZ LONDONO, "Analyse du flux des données dans un système parallèle", *Masters (DEA) dissertation*, Institut National Polytechnique de Grenoble, June 1993.

VASY team as a development tool for compiler construction [2].

In 2004, we released a new version 2.4 of TRAIAN, which corrects two bugs of the previous version 2.3 issued in 2003. In addition, we ported the TRAIAN compiler to the MAC OS X and recent LINUX operating systems.

We also continued to work on the compilation of the process part of E-LOTOS and LOTOS NT, which is a difficult problem as these languages combine concurrency, quantitative time, and exceptions. To deal with these problems progressively, we chose to focus first on the sequential processes present in E-LOTOS and LOTOS NT. In 2002, we designed a formalism named NTIF (*New Technology Intermediate Form*) to be used as an intermediate language for compiling and verifying E-LOTOS and LOTOS NT processes.

NTIF allows to specify extended automata parameterized by typed variables. Each transition is labeled with an action (which allows communication with the environment according to the rendezvous semantics of process algebras) and a sequential code fragment to read and/or write variables. Compared to classical "*condition/action*" (or "*guarded commands*") automata, NTIF provides high level control structures (statements "*case*", "*if-then-else*", "*while*", etc.); this avoids the introduction of spurious intermediate states and transitions, as well as the duplication of boolean conditions, an important source of errors [4].

Since 2003, NTIF also features quantitative time concepts, in the form of a "*wait*" operator that lets a given amount of time elapse, timing tags on actions to express deadline and urgency, and a construct to capture the time elapsed between the instant an action is enabled and the instant it actually occurs.

In 2004, we revised and simplified the semantics of timed NTIF automata, by observing that properties inherent to time (namely time additivity, time determinism, and maximal progress) can be decoupled from the structural semantics definition. To this aim, we defined a time equivalence relation on *Timed Labeled Transition Systems*, the underlying model of NTIF. This lead to structural operational semantics rules that are lightweight extensions of the untimed rules. Besides NTIF, this approach could be used to simplify the (generally complex) semantics of many timed process algebras.

5.2.4 Source-Level Translations between Process Algebras

Participants: Hubert Garavel, Frédéric Lang, Gwen Salaün.

Although process algebras are, from a technical point of view, the best formalism to describe concurrent systems, they are not used as widely as it 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 to this fragmentation.

Because of this, process algebras other than LOTOS do not benefit from the advanced functionalities provided by the CADP toolbox. In 2004, we started to address this problem by designing source-level translators from various process algebras into LOTOS.

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 and which is used by the LETI laboratory to describe complex, asynchronous circuits at a high abstraction level.

G. Salaün undertook the development of a translator from CHP to LOTOS (currently, 2,130 lines of SYNTAX compiler-generator code, 1,970 lines of LOTOS NT code, and 350 lines of C code). When completed, this translator should allow CHP designs to be verified using CADP before being turned into asynchronous circuits using TAST.

Besides the case of CHP, G. Salaün started studying translators for other process algebras, including CSP/FDR2 and FSP.

5.3 Case Studies and Practical Applications

Participants: Damien Bergamini, David Champelovier, Aurore Collomb, Hubert Garavel, Christophe Joubert, Frédéric Lang, Radu Mateescu, Gwen Salaün, Wendelin Serwe.

In 2004, the VASY team also worked on the following case studies:

- We collaborated with Gwen Salaün (formerly at University "La Sapienza", Rome) on the verification of negotiating WEB services involving clients and providers, who try to agree on some information (e.g., prices) [SFC04]. In this case study, we used our latest prototype of CÆSAR (see § 5.2.2) to verify large configurations.
- In the framework of the INRIA/LETI collaboration (see § 7.1), we studied the suitability of LOTOS and CADP for the verification of asynchronous circuits. We developed a LOTOS specification (about 2,000 lines of code) for an asynchronous circuit, designed by the LETI and TIMA laboratories, which implements the DES (*Data Encryption Standard*) algorithm. We successfully verified several correctness properties about the control flow of the circuit, such as absence of deadlocks, correct number of iterations, etc.
- We studied a cache coherence protocol for a multiprocessor architecture specified in LOTOS by Massimo Zendri (BULL) in 1999. This protocol is based on a remote directory used by concurrent agents. Using a simple compositional verification approach, we could generate the state space for 5 agents, but failed to handle larger configurations, because the remote directory process became too large (1 million states, 40 million transitions for 5 agents) to be generated in isolation for more than 5 agents. The automated interface generation feature of EXP.OPEN 2.0 (see § 5.1.5) allowed us to constrain tightly the remote directory process (which never exceeded 60 states) and to generate the state space for 7 agents (1 million states and 7 million transitions).
- We collaborated with Holger Hermanns and Sven Johr (Saarland University) to generate a stochastic model of a distributed mutual exclusion algorithm. The model was obtained from the parallel composition using EXP.OPEN 2.0 of "sequential" stochastic models

[SFC04] G. SALAÜN, A. FERRARA, A. CHIRICHIELLO, "Negotiation among Web Services using LOTOS/CADP", *in: Proceedings of the European Conference on Web Services ECOWS'04 (Erfurt, Germany)*, L.-J. Zhang (editor), *Lecture Notes in Computer Science*, *3250*, Springer Verlag, p. 198–212, September 2004. Extended version available as Technical Report 13-04 of Università di Roma "La Sapienza" (DIS department).

corresponding to each concurrent process. Using the CADP toolbox, all configurations with up to 5 processes could be generated without problem. For the configuration with 6 processes, we first generated, using DISTRIBUTOR on a cluster of 8 machines, a PBG model (see § 5.1.6), which was unfortunately too large (224 million states, 1,300 million transitions, 12 Gbytes) to fit on standard 32-bit file systems; therefore, we used the partial order reduction for stochastic systems implemented in EXP.OPEN 2.0 (see Section 5.1.5) together with DISTRIBUTOR; using a cluster of 11 machines, we managed to generate a much smaller, yet equivalent, state space (44 million states, 80 million transitions) in less than 7 minutes.

- We continued the work undertaken in collaboration with Grégory Batt and Hidde de Jong (HELIX team of INRIA Rhône-Alpes) for connecting the GNA (*Genetic Network Analyzer*) tool developed by HELIX with CADP in order to verify temporal properties of genetic regulatory networks.

  GNA provides a simulator of qualitative models of genetic regulatory networks in the form of piecewise-linear differential equations. The connection is performed by the GNA2BCG translator from the graph format produced by GNA to the BCG format. The resulting BCG graph can be simplified by eliminating instantaneous states using abstraction and reduction modulo branching equivalence, and inspected visually by using the BCG_EDIT tool of CADP. Then, various temporal properties concerning the behavior of the regulatory network (evolution of protein concentrations, reachability of equilibrium states, etc.) can be verified using the EVALUATOR 3.0 model checker.

  In 2004, we improved the translation between the graphs produced by GNA and those accepted by CADP in order to preserve strong equivalence, and we defined the CTL operators, which are useful for expressing properties of genetic regulatory networks, as a library (50 lines of regular alternation-free μ-calculus) in EVALUATOR 3.0. The joint use of GNA and CADP for model checking genetic regulatory networks was subject to a publication [18].

- In the framework of the FORMALFAME contract (see § 6.2), we performed a comparative study of the MURφ [Dil96] and CADP verification tools. For this study, BULL selected as a benchmark a cache coherence protocol for multiprocessor architectures present in the MURφ distribution (under the name "CACHE3").

  We translated manually the MURφ code (1,000 lines) into LOTOS (1,800 lines of LOTOS). Both languages do not provide the same level of description: MURφ requires a sequential abstraction in which the entire protocol is viewed as a single sequential process handling global data, whereas LOTOS allows a more detailed view, closer to the actual implementation, in which the protocol consists of several distributed processes having their own local memories.

  We analysed using CADP several configurations of the protocol with one memory containing up to 2 different data values, one cache line, and up to 3 processors.

  For the simplest configuration (one data value and one processor), the corresponding state space (7,694 states, 10,242 transitions) could be generated in a few seconds.

[Dil96] D. DILL, "The Murφ Verification System", *in: Proceedings of the 8th International Conference on Computer-Aid ed Verification CAV'96*, R. Alur, T. Henzinger (editors), *Lecture Notes in Computer Science*, *1102*, Springer Verlag, p. 390–393, July 1996.

For a more complex configuration (two data values and three processors), the compositional verification tools of CADP allowed to generate the corresponding state space (about 3 million states, 18 million transitions) in about 40 minutes on a standard PC. On the same example, MURφ, when used without its symmetry reduction feature, took 58 minutes to generate a state space of 760,000 states; enabling symmetry reduction led to a smaller state space (about 16,000 states) that could be generated in 3 minutes.

On this example, the state spaces generated using CADP are larger than those of MURφ, which corresponds to the fact that, using LOTOS, we modeled the cache coherence protocol at a finer degree of granularity.

On the opposite, the correctness properties that can be verified using MURφ are restricted to mere state invariants on the variables of the MURφ specification ("white box" approach), whereas CADP allows sophisticated properties relying the input and output events of the protocol ("black box" approach); for instance, we specified the user-level view of memory consistency as a μ-calculus formula, which we evaluated using CADP.

- We collaborated with Judi Romijn and Stefan Vorstenbosch (Eindhoven University of Technology) on the verification of the Net Update Protocol that is part of the IEEE P1394.1 draft standard.

  In his Master thesis (May 2004), Stefan Vorstenbosch developed successive LOTOS specifications for this protocol, which he tried to verify using the CADP toolbox.

  We contributed to the specification task by suggesting to model broadcast (i.e., n-to-n communications) using LOTOS n-ary rendezvous. This allowed a significant reduction in the size of the Petri nets generated by CÆSAR.

  We contributed to the verification task by using our latest prototype of CÆSAR (see § 5.2.2), our compositional verification tools (see § 5.1.5), and our distributed verification tools (see § 5.1.6) to study complex configurations of the protocol.

  Using CÆSAR directly on a single machine, we managed to generate the state space for one net update (nearly 21,000 states and 77,000 transitions) and two net updates (nearly 2 million states and 10 million transitions). Using a compositional approach, we managed to generate and minimize a state space valid for any number of net updates (nearly 5 million states and 61 million transitions). We discovered the presence of deadlocks in the protocol as soon as more than one net update was allowed. Based on the counter-example traces reported by us, Stefan Vorstenbosch removed some (but not all) deadlocks from the specification.

  In September 2004, we studied an updated specification provided to us by Judi Romijn. Using abstractions, compositional, and distributed verification, we generated a tractable state space (nearly 8 million states and 88 million transitions) despite the fact that some intermediate state spaces were larger (28 million states and 487 million transitions). Unfortunately, this state space still contained deadlocks, which are under investigation in Eindhoven.

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

- the verification of control properties of asynchronous circuits [BBM+03],
- the automatic verification of the Root Contention Protocol of the IEEE 1394 "FIREWIRE" bus [DKN03],
- the formal specification and verification of a fair payment protocol [CD04],
- the formal specification and verification of middleware behavior for the CORBA transaction service [RC04], and
- the parameterized specification and verification of the Chilean electronic invoice system [ABM04].

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

- the MCRL.OPEN tool, developed by Jaco van de Pol (CWI, Amsterdam), which compiles descriptions written in the μCRL process algebra into C code that complies with the OPEN/CÆSAR programming interface, thus allowing all CADP tools that operate on the fly to be applied to μCRL descriptions; in the framework of the SENVA collaboration (see § 7.2 below), H. Garavel helped to improve the memory performance of MCRL.OPEN;
- an abstract interpretation toolkit for μCRL [vdPE04a], which uses CADP to verify modal labeled transition systems generated from μCRL descriptions [vdPE04b].

[BBM+03] D. BORRIONE, M. BOUBEKEUR, L. MOUNIER, M. RENAUDIN, A. SIRIANNI, "Validation of Asynchronous Circuit Specifications using IF/CADP", *in: Proceedings of the International Conference on Very Large Scale Integration of System-on-Chip VLSI-SoC 2003 (Darmstadt, Germany)*, M. Glesner, R. A. da Luz Reis, H. Eveking, V. J. Mooney, L. S. Indrusiak, P. Zipf (editors), p. 86–91, Darmstadt, December 2003.

[DKN03] C. DAWS, M. Z. KWIATKOWSKA, G. NORMAN, "Automatic Verification of the IEEE 1394 Root Contention Protocol with KRONOS and PRISM", *International Journal on Software Tools for Technology Transfer (STTT) 5*, 2–3, 2003, p. 109–137.

[CD04] J. G. CEDERQUIST, M. T. DASHTI, "Formal Analysis of a Fair Payment Protocol", *Technical Report number SEN-R0410*, CWI, Amsterdam, The Netherlands, July 2004.

[RC04] N. S. ROSA, P. R. F. CUNHA, "A Software Architecture-Based Approach for Formalising Middleware Behaviour", *Electronic Notes in Theoretical Computer Science*, *108*, p. 39–51, December 2004.

[ABM04] I. ATTALI, T. BARROS, E. MADELAINE, "Parameterized Specification and Verification of the Chilean Electronic Invoices System", *in: Proceedings of the XXIV International Conference of the Chilean Computer Science Society SCCC'04 (Arica, Chili)*, Society for Computer Simulation International, IEEE, p. 14–25, November 2004.

[vdPE04a] J. VAN DE POL, M. V. ESPADA, "An Abstract Interpretation Toolkit for μCRL", *in: Proceedings of the 9th International Workshop on Formal Methods for Industrial Critical Systems FMICS'04 (Linz, Austria)*, J. Bicarregui, A. Butterfield (editors), *Electronic Notes in Theoretical Computer Science*, September 2004.

[vdPE04b] J. VAN DE POL, M. V. ESPADA, "Modal Abstractions in μCRL", *in: Proceedings of the 10th International Conference on Algebraic Methodology and Software Technology AMAST'2004 (Stirling, Scotland, UK)*, C. Rattray, S. Maharaj, C. Shankland (editors), *Lecture Notes in Computer Science*, *3116*, Springer Verlag, p. 409–425, July 2004. Also available as CWI Technical Report SEN-R0401.

6 Contracts and Grants with Industry

6.1 The IST ArchWare European Contract

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

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

The aim of ARCHWARE is to build an integrated environment for architecting evolvable software systems with functional and performance requirements. Based on a software architecture description language, this environment will offer functionalities to define architectural styles specific to various activity domains, as well as engineering tools for analyzing architectural descriptions. The role of VASY in ARCHWARE concerns the description and verification of functional properties.

In 2004, we continued the development of the model checker for the AAL (*Architecture Analysis Language*) fragment dedicated to the description of behavioral properties of software architectures, and we contributed to the integration of this tool within the ARCHWARE environment (see § 5.1.4).

The description of ARCHWARE and of its current achievements was subject to a publication [26].

6.2 The FormalFame Contract

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

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

A first phase of this collaboration took place from 1995 to 1998 in the framework of the DYADE joint venture between BULL and INRIA. Two case studies were successfully tackled: the POWERSCALE bus arbitration protocol [CGM+96] and the POLYKID multiprocessor architecture [9].

[CGM+96] 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 Verifi-*

The feasibility of the proposed approach was established and BULL expressed its interest in pursuing the collaboration for its new architectures.

Since October 1998, we have been working on FAME, the CC-NUMA multiprocessor architecture developed by BULL for its NOVASCALE series of high-performance servers based on INTEL ITANIUM 64 bits processors. Initially informal, this collaboration was officialized in 1999 as a DYADE action named FORMALFAME, which lasted until the end of DYADE in March 2001. The collaboration went on under the form of a BULL-INRIA contract, for which we kept the name FORMALFAME. In 2004, the collaboration was extended until March 2005 by a followup contract named FORMALFAME PLUS.

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

Compared to the previous years, the nature of the BULL/INRIA collaboration evolved, for at least two reasons: first, the NOVASCALE servers went to market successfully, meaning that the development of the cache coherence protocol is over; second, BULL has acquired a sufficient autonomy in formal methods to maintain by itself the LOTOS specifications developed for ILU and PRR. In this context, the contributions of INRIA in 2004 were the following:

- expertise regarding some involved aspects of LOTOS and CADP,
- support for migration to the latest versions of CADP,
- compatibility patches allowing CADP to run on the old version of LINUX (REDHAT 7.3) for which CADENCE's tools are available,
- enhancements of CADP tools to address issues detected in previous years, and
- two case-studies on the formal verification of cache coherency protocols (see § 5.3).

7 Other Grants and Activities

7.1 National Collaborations

In 2004, our preliminary collaboration with the LETI laboratory of CEA-Grenoble was formalized in the framework of the INRIA/LETI co-operation agreement. VASY is involved in a research action about asynchronous circuits, GALS (*Globally Asynchronous Locally Synchronous*) architectures, NOC (*Network on Chip*), and SOC (*System on Chip*), together with LETI scientists (Edith Beigné, François Bertrand, and Pascal Vivet). The TIMA laboratory

cation FORTE/PSTV'96 (Kaiserslautern, Germany), R. Gotzhein, J. Bredereke (editors), IFIP, Chapman & Hall, p. 435–450, October 1996. Full version available as INRIA Research Report RR-2958, `http://www.inria.fr/rrrt/rr-2958.html`.

(Dominique Borrione, Menouer Boubekeur, and Marc Renaudin) also contributes to this research action. The goal is to build software tools for the design of asynchronous circuits; such circuits are very promising in terms of high-performance, low-power, low-emission, and modularity; however, their design is difficult due to the complexity inherent to asynchronous concurrency. In 2004, our work focused on the verification of an asynchronous circuit implementing the DES encryption standard (see § 5.3) and on the development of a translator to interconnect the verification tools developed by VASY and the hardware synthesis tools developed at TIMA (see § 5.2.4).

Together with the project-team OASIS of INRIA Sophia-Antipolis (Isabelle Attali, Tomas Barros, Rabea Boulifa, Eric Madelaine, and Bernard Serpette), the LTCI team of ENST-Paris (Elie Najm and Sylvie Vignes), and the SVF team of the FERIA-LAAS laboratory (Bernard Berthomieu and François Vernadat), VASY is part of the national action FIACRE – ACI *Sécurité Informatique* started in 2004 (see `http://www-sop.inria.fr/oasis/fiacre`). F. Lang played a major role in launching the FIACRE action.

In 2004, we collaborated with several INRIA teams:

- APACHE and SARDES (Rhône-Alpes): use of the I-CLUSTER, IDPOT, and ION clusters to experiment parallel and distributed verification algorithms (see § 5.1.6);
- HELIX (Rhône-Alpes): applications of model checking to biological systems (Grégory Batt and Hidde de Jong);
- OASIS (Sophia-Antipolis): collaboration in the framework of the FIACRE action;
- VERTECS (Rennes): enhancements to the TGV tool (Thierry Jéron).

Beyond INRIA, we had scientific relations with the following team:

- LAMI laboratory (Evry): coordination of distributed processes with behavioral interfaces (Pascal Poizat).

Finally, H. Garavel is an INRIA representative at CNRT-AE (*Centre National de Recherche Technologique – Aéronautique et Espace*) and at the national project NUM@TEC AUTOMOTIVE.

7.2 International Collaborations

In 2004, the VASY team of INRIA and the SEN2 team of CWI launched a joint research team on safety-critical systems (see `http://www.inrialpes.fr/vasy/senva`). This initiative is supported by INRIA's European and International Affairs Department.

The VASY team is member of the FMICS (*Formal Methods for Industrial Critical Systems*) working group of ERCIM (see `http://www.inrialpes.fr/vasy/fmics`). From July 1999 to July 2001, H. Garavel chaired this working group. Since July 2002, he is member of the FMICS Board, in charge of dissemination actions. On April 20–22, 2004, R. Mateescu organized in Aix-les-Bains an FMICS seminar devoted to the preparation of a "Formal Methods Handbook".

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

We maintain scientific relations with several international universities and research centers. In addition to our partners in aforementioned contractual collaborations, we had scientific exchanges in 2004 with:

- Eindhoven University of Technology (Judi Romijn and Stefan Vorstenboch),
- University of Kent at Canterbury (Clara Benac Earle),
- University of Rome "La Sapienza" (Gwen Salaün),
- Radboud University Nijmegen (Tim Willemse),
- Saarland University (Holger Hermanns and Sven Johr),
- University of Twente (Axel Belinfante).

7.3 Visits and Invitations

In the framework of the ARCHWARE project (see § 6.1) and the SENVA collaboration (see § 7.2), we invited the following visitors:

- Jaco van de Pol (CWI, Amsterdam), visited us on May 3–4, 2004. He gave a demonstration of the μCRL toolset.
- Stefan Blom, Wan Fokkink, Bert Lisser, and Jaco van de Pol (CWI, Amsterdam), Judi Romijn (University of Eindhoven), Holger Hermanns and Sven Johr (Saarland University) attended the SENVA 2004 workshop, held in Allevard-Les-Bains on June 21–24, 2004:
  - Stefan Blom gave a talk entitled "*Bisimulation Reduction Using Signature Based Partition Refinement*".
  - Wan Fokkink gave a talk entitled "*Verification of a Sliding Window Protocol in* μCRL".
  - Holger Hermanns gave three talks entitled "*Stochastic Modelling and Analysis*", "*Model Checking Markov Chains and Decision Processes*", and "*MODEST: A Model Description Language for Stochastic Timed Systems*".
  - Sven Johr gave a talk entitled "*Turning Interactive Markov Chains to Continuous-Time Markov Decision Processes*".
  - Bert Lisser gave a talk entitled "*Reducing state spaces by transforming linear process equations*" and gave a demonstration of the JSIM-BFS simulator.
  - Jaco van de Pol gave a talk entitled "*Abstract Model Checking for Process Algebra*".
  - Judi Romijn gave two talks entitled "*Guiding Spin Simulation*" and "*Correcting IEEE 1394.1 FireWire Net Update*".

- Nando Gallo (CPR, Pisa), Brian Warboys (University of Manchester), Flavio Oquendo (Université de Savoie), and Regis Dindeleux (THESAME) visited us on September 14–16, 2004 for an ARCHWARE project meeting that we organized.
- Stefan Blom visited us on September 21–25, 2004. He gave a talk entitled "*Model Checking Archive: A File Format for Distributed State Space Generation Using a Central Storage Location*".

8 Dissemination

8.1 Software Dissemination and Internet Visibility

The VASY team distributes two main software tools: the CADP toolbox (see § 4.1) and the TRAIAN compiler (see § 4.2). In 2004, the main facts are the following:

- We prepared and distributed 21 successive beta-versions (2002-x, ..., 2002-z, 2003-a, ..., 2003-r) of CADP.
- The number of license contracts signed for CADP increased from 307 to 330.
- We were requested to grant CADP licenses for 679 different computers in the world.
- The distribution of the TRAIAN compiler continued and a new version 2.4 of TRAIAN (see § 5.2.3) was released on June 8, 2004.
- The TRAIAN compiler was downloaded by 76 different sites.

The VASY WEB site was regularly updated with scientific contents, announcements, publications, etc. Most notably, the following contents were produced in 2004:

- The CADP WEB site was entirely redesigned. More readable and better structured, the new WEB site also features CADP latest news and enhancements (see `http://www.inrialpes.fr/vasy/cadp`).
- The CADP *Frequently Asked Questions* page was enriched with 18 new entries (see `http://www.inrialpes.fr/vasy/cadp/faq.html`).

8.2 Program Committees

In 2004, the members of VASY assumed the following responsibilities:

- H. Garavel was, together with John Hatcliff (Kansas State University), responsible for a special issue of the TCS (*Theoretical Computer Science*) journal, to appear, which gathers the best theory-oriented papers of TACAS'2003.
- H. Garavel was, together with John Hatcliff (Kansas State University), responsible for a special issue of the STTT (*Software Tools for Technology Transfer*) journal, to appear, which gathers the best software-oriented papers of TACAS'2003.

- R. Mateescu was a program committee member of VVEIS'2004 (*2nd International Workshop on Verification and Validation of Enterprise Information Systems*, Porto, Portugal, April 13, 2004).
- R. Mateescu was a program committee member of EWSA'2004 (*1st European Workshop on Software Architectures*, St Andrews, Scotland, May 21–22, 2004).
- H. Garavel was a program committee member of PDMC'2004 (*3rd International Workshop on Parallel and Distributed Methods in Verification*, London, England, August 31–September 3, 2004) and contributed to paper selection for a related special issue of the FMSD (*Formal Methods in System Design*) journal.
- R. Mateescu was a program committee member of FMICS'2004 (*9th International Workshop on Formal Methods for Industrial Critical Systems*, Linz, Austria, September 20–21, 2004).

8.3 Lectures and Invited Conferences

In 2004, we gave talks in several international conferences and workshops (see bibliography below). Additionally:

- H. Garavel participated to the scientific committee of the 11th INRIA-Industry meeting on software engineering (INRIA Rocquencourt) on January 27, 2004. At this meeting, he also gave a talk entitled "*Analyse et vérification automatique de systèmes asynchrones*", and F. Lang and R. Mateescu demonstrated the CADP toolbox.
- A. Collomb gave a talk entitled "*Vers des systèmes asynchrones sûrs*" at the LISTIC laboratory (Annecy) on March 18, 2004.
- H. Garavel gave an invited talk entitled "*Almost Ten Years of Process Algebras and Model Checking for Multiprocessor Architectures*" at X-TACAS, the 10th Anniversary of TACAS (Barcelona, Spain) on March 27–28, 2004.
- R. Mateescu gave a keynote presentation entitled "*A Generic Framework for Model Checking Software Architectures*" at the 2nd International Workshop on Verification and Validation of Enterprise Information Systems VVEIS'2004 (Porto, Portugal) on April 13–14, 2004.
- H. Garavel gave a talk entitled "*Almost Ten Years of Process Algebras and Model Checking for Multiprocessor Architectures*" at the PAM (*Process Algebra Meeting*) held at CWI (Amsterdam, The Netherlands) on May 12, 2004.
- The annual SENVA seminar was held in Allevard-Les-Bains on June 21–24, 2004. The list of talks is available from `http://www.inrialpes.fr/vasy/senva/workshop2004`.
- F. Lang gave two talks entitled "*Compositional Verification Using* CADP *of the* SCALAGENT *Deployment Protocol for Software Components*" and "*Activités de recherche du projet* VASY" at the kick-off meeting of the FIACRE national action (INRIA Sophia-Antipolis) on November 22–23, 2004.

- C. Joubert gave a talk entitled "*Analyse d'espaces d'états par résolution distribuée de systèmes d'équations booléennes*" at the "*Journée des doctorants de l'Ecole Doctorale Mathématiques, Informatique, Sciences et Technologies de l'Information*" MSTII'2004 (Grenoble) on December 2, 2004.
- G. Salaün gave an invited talk entitled "*Specification and Verification of Asynchronous Systems*" at the Indo-French Seminar on Information Technology organized by IFCPAR, the Indo-French Centre for the Promotion of Advanced Research (Pune, India) on December 10, 2004.

8.4 Teaching Activities

The VASY team is a host team for:

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

In 2004:

- C. Joubert gave the course on "*Algorithmique et programmation impérative*" to computer science students at Université Joseph Fourier, Grenoble (license 2nd year, 37.5 hours).
- C. Joubert gave the course on "*Architecture Logicielle et Matérielle*" to the 2nd year students of IUP MIAGE at Université Joseph Fourier, Grenoble (18 hours).
- C. Joubert gave the course on "*Géométrie euclidienne, analyse approfondie et introduction à l'algébre linéaire*" to computer science students at Université Joseph Fourier, Grenoble (license 1st year, 20 hours).
- R. Mateescu and W. Serwe gave the course on "*Temps Réel*" to the 3rd year students of ENSIMAG (21 hours).
- H. Garavel was a jury member of Jun Pang's PhD thesis entitled "*Formal Verification of Distributed Systems*", defended at the Free University of Amsterdam (The Netherlands) on October 26, 2004.
- H. Garavel was a jury member of Nestor Cataño Collado's PhD thesis entitled "*Méthodes formelles pour la vérification des programmes Java*", defended at Université Paris 7 on November 4, 2004.
- R. Mateescu was a jury member of Rabea Boulifa's PhD thesis entitled "*Génération de modèles comportementaux des applications réparties*", defended in Sophia-Antipolis on December 15, 2004.
- R. Mateescu was a member of the "commission de spécialistes" at Université de Savoie (section 27).

9 Bibliography

Reference Publications by the Team

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

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

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

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

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

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

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

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

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

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

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

[12] 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.

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

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

Journal Articles and Book Chapters

[15] H. Garavel, W. Serwe, "State Space Reduction for Process Algebra Specifications", *Theoretical Computer Science*, 2005, to appear.

[16] F. Lang, "Explaining the Lazy Krivine Machine Using Explicit Substitution and Addresses", *Journal of Higher-Order and Symbolic Computation, special issue on Krivine's machine*, 2005, to appear.

[17] R. Mateescu, "CAESAR_SOLVE: A Generic Library for On-the-Fly Resolution of Alternation-Free Boolean Equation Systems", *Springer International Journal on Software Tools for Technology Transfer (STTT)*, 2005, to appear.

Publications in Conferences and Workshops

[18] G. Batt, D. Bergamini, H. de Jong, H. Garavel, R. Mateescu, "Model Checking Genetic Regulatory Networks using GNA and CADP", *in : Proceedings of the 11th International SPIN Workshop on Model Checking of Software SPIN'2004 (Barcelona, Spain)*, S. Graf, L. Mounier (editors), *Lecture Notes in Computer Science*, *2989*, Springer Verlag, p. 156–161, April 2004.

[19] D. Bergamini, N. Descoubes, C. Joubert, R. Mateescu, "BISIMULATOR: A Modular Tool for On-the-Fly Equivalence Checking", *in : Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2005 (Edinburgh, Scotland, UK)*, N. Halbwachs, L. Zuck (editors), *Lecture Notes in Computer Science*, Springer Verlag, April 2005. to appear.

[20] H. Garavel, R. Mateescu, "SEQ.OPEN: A Tool for Efficient Trace-Based Verification", *in : Proceedings of the 11th International SPIN Workshop on Model Checking of Software SPIN'2004 (Barcelona, Spain)*, S. Graf, L. Mounier (editors), *Lecture Notes in Computer Science*, *2989*, Springer Verlag, p. 150–155, April 2004.

[21] H. Garavel, W. Serwe, "State Space Reduction for Process Algebra Specifications", *in : Proceedings of the 10th International Conference on Algebraic Methodology and Software Technology AMAST'2004 (Stirling, Scotland, UK)*, C. Rattray, S. Maharaj, C. Shankland (editors), *Lecture Notes in Computer Science*, *3116*, Springer Verlag, p. 164–180, July 2004.

[22] B. Jeannet, W. Serwe, "Abstracting Call-Stacks for Interprocedural Verification of Imperative Programs", *in : Proceedings of the 10th International Conference on Algebraic Methodology and Software Technology AMAST'2004 (Stirling, Scotland, UK)*, C. Rattray, S. Maharaj, C. Shankland (editors), *Lecture Notes in Computer Science*, *3116*, Springer Verlag, p. 258–273, July 2004.

[23] C. Joubert, R. Mateescu, "Distributed On-the-Fly Equivalence Checking", *in : Proceedings of the 3rd International Workshop on Parallel and Distributed Methods in Verification PDMC'2004 (London, UK)*, L. Brim, M. Leucker (editors), *Electronic Notes in Theoretical Computer Science*, Elsevier, 2004.

[24] C. Joubert, R. Mateescu, "Distributed Local Resolution of Boolean Equation Systems", *in : Proceedings of the 13th Euromicro Conference on Parallel, Distributed and Network-Based Processing PDP'2005 (Lugano, Switzerland)*, F. Tirado, M. Prieto (editors), IEEE Computer Society, February 2005. to appear.

[25] R. Mateescu, "Model Checking for Software Architectures", *in : Proceedings of the 1st European Workshop on Software Architecture EWSA'2004 (St Andrews, Scotland, UK)*, F. Oquendo, B. Warboys, R. Morrison (editors), *Lecture Notes in Computer Science*, *3047*, Springer Verlag, p. 219–224, May 2004.

[26] F. Oquendo, B. Warboys, R. Morrison, R. Dindeleux, F. Gallo, H. Garavel, C. Occhipinti, "ArchWare: Architecting Evolvable Software", *in : Proceedings of the 1st European Workshop on Software Architecture EWSA'2004 (St Andrews, Scotland, UK)*, F. Oquendo, B. Warboys, R. Morrison (editors), *Lecture Notes in Computer Science*, *3047*, Springer Verlag, p. 257–271, May 2004. Invited paper.

Research Reports and Internal Publications

[27] D. Bergamini, D. Champelovier, N. Descoubes, H. Garavel, R. Mateescu, W. Serwe, "ArchWare Architecture Analysis Tool by Model-Checking", *Project Deliverable number D3.6b*, IST Project 2001-32360 "ArchWare", June 2004.

[28] D. Bergamini, D. Champelovier, N. Descoubes, H. Garavel, R. Mateescu, W. Serwe, "Final ArchWare Architecture Analysis Tool by Model-Checking", *Project Deliverable number D3.6c*, IST Project 2001-32360 "ArchWare", December 2004.

Miscellaneous

[29] W. Fokkink, H. Garavel, J. van de Pol, "CWI and INRIA join Forces on Safety Critical Systems", Ercim News 58, July 2004.

[30] Vasy, "Bcg_Merge Manual Page", December 2004, `http://www.inrialpes.fr/vasy/cadp/man/bcg_merge.html`.

[31] Vasy, "Bcg_Steady Manual Page", December 2004, `http://www.inrialpes.fr/vasy/cadp/man/bcg_steady.html`.

[32] Vasy, "Bcg_Transient Manual Page", December 2004, `http://www.inrialpes.fr/vasy/cadp/man/bcg_transient.html`.

[33] VASY, "Bisimulator Manual Page", December 2004, `http://www.inrialpes.fr/vasy/cadp/man/bisimulator.html`.

[34] VASY, "Caesar.Bdd Manual Page", July 2004, `http://www.inrialpes.fr/vasy/cadp/man/caesar.bdd.html`.

[35] VASY, "Determinator Manual Page", December 2004, `http://www.inrialpes.fr/vasy/cadp/man/determinator.html`.

[36] VASY, "Exp.Open Version 2 Manual Page", August 2004, `http://www.inrialpes.fr/vasy/cadp/man/exp.open.html`.

[37] VASY, "Projector Version 2 Manual Page", December 2004, `http://www.inrialpes.fr/vasy/cadp/man/projector.html`.