# INRIA, Evaluation of Theme ComC

Project-team VASY

April 2007

**Project-team title: VASY**

**Scientific leader: Hubert Garavel**

**Research center: INRIA Rhône-Alpes**

**Common project-team with:**

**Personnel (June 2003)**

| | Industry | INRIA | CNRS | University | **Total** |
|---|---|---|---|---|---|
| DR (1) / Professors | | 1 | | | **1** |
| CR (2) / Assistant Professors | | 2 | | | **2** |
| Permanent Engineers (3) | 1 | | | | **1** |
| Temporary Engineers (4) | | 4 | | | **4** |
| PhD Students | | 1 | | | **1** |
| Post-Doc. | | | | 1 | **1** |
| **Total** | **1** | **8** | | **1** | **10** |
| External Collaborators | | | | | |
| Visitors (> 1 month) | | | | | |

(1) "Senior Research Scientist (Directeur de Recherche)"
(2) "Junior Research Scientist (Chargé de Recherche)"
(3) "Civil servant (CNRS, INRIA, ...)"
(4) "Associated with a contract (Ingénieur Expert or Ingénieur Associé)"

**Personnel (April 2007)**

| | Industry | INRIA | CNRS | University | **Total** |
|---|---|---|---|---|---|
| DR / Professors | | 1 | | | **1** |
| CR / Assistant Professor | | 3 | | | **3** |
| Permanent Engineer | 1 | | | | **1** |
| Temporary Engineer | | 5 | | | **5** |
| PhD Students | 2 | 1 | | | **3** |
| Post-Doc. | | 1 | | | **1** |
| **Total** | **3** | **11** | | | **14** |
| External Collaborators | | | | | |
| Visitors (> 1 month) | | 1 | | | **1** |

**Changes in staff**

| DR / Professors<br>CR / Assistant Professors | Misc. | INRIA<br>1 | CNRS | University | total |
|---|---|---|---|---|---|
| Arrival | | 1 CR | | | 1 |
| Leaving | | | | | |

## Current composition of the project-team on April 2007:

- David Champelovier (Inria temporary engineer)
- Nicolas Coste (PhD student funded by ST Microelectronics)
- Marlyse Felici (Inria administrative assistant)
- Hubert Garavel (DR2 Inria, project-team leader)
- Rémi Hérilier (Inria temporary engineer)
- Holger Hermanns (visiting professor from Saarland University) – in 2007
- Romain Lacroix (Inria temporary engineer)
- Frédéric Lang (CR1 Inria)
- Etienne Lantreibecq (ST Microelectronics permanent engineer)
- Radu Mateescu (CR1 Inria)
- Olivier Ponsini (Inria post-doc)
- Sylvain Robert (Inria temporary engineer)
- Wendelin Serwe (CR2 Inria)
- Jan Stoecker (PhD student funded by Inria)
- Damien Thivolle (Inria MSc internship)
- Marie Vidal (Inria temporary engineer)
- Meriem Zidouni (PhD student funded by Bull)

## Current position of former project-team members (including PhD students during the June 2003 – April 2007 period):

- Damien Bergamini (former Inria temporary engineer): R&D engineer at Bull (Echirolles).
- Aurore Collomb (former Inria post-doc): Engineer at a French company.
- Nicolas Descoubes (former Inria temporary engineer): R&D engineer at Mentor Graphics (Montbonnot).
- Christophe Joubert (former PhD student): Post-doc at University of Valencia (Spain).
- Abdul Malik Khan (former MSc internship): PhD student at Enst Brest.

- Solofo Ramangalahy (former Bull permanent engineer): R&D engineer at Bull (Echirolles).
- Gwen Salaün (former Inria post-doc): Post-doc at University of Valencia (Spain).
- Frédéric Tronel (former Inria temporary engineer): Assistant professor at Ifsic / University of Rennes I.

### Last INRIA enlistments

- 2004, Wendelin Serwe, CR2

### Other comments:

Due to difficulties for Vasy to obtain PhD students in Grenoble (a problem pointed out during the previous evaluation in 2003), Vasy decided to join the Lig (*Laboratoire d'Informatique de Grenoble*, launched in January 2007) in an attempt to draw closer links with the local universities.

Between April 2005 and November 2006, R. Mateescu has been hosted by the Lip laboratory at Ecole Nationale Supérieure de Lyon. Since January 2007, R. Mateescu is hosted by the Le2i laboratory in Dijon.

# 1 Work progress

## 1.1 Keywords

asynchronous parallelism, automaton, bisimulation, code generation, communication protocol, compilation, compiling, concurrency, critical application, distributed application, distributed system, formal methods, formal specification, modeling, mu-calculus, process algebra, program verification, real-time, software engineering, synchronization, system design, temporal logic.

## 1.2 Context and overall goal of the project

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

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

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

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

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

- Models provide mathematical representations for parallel programs and related verification problems. Examples of models are automata, networks of communicating automata, Petri nets, binary decision diagrams, boolean equation systems, etc. From a theoretical point of view, research on models seeks for general results, independently from any particular description language.

- In practice, models are often too elementary to describe complex systems directly (this would be tedious and error-prone). Higher level formalisms are needed for this task, as well as compilers that translate high level descriptions into models suitable for verification algorithms.

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

### 1.2.1 Models and Verification Techniques

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

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

- *Behavioral properties* express the intended functioning of the system in the form of automata (or higher level descriptions, which are then translated into automata). In 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 $\mu$-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 $\mu$-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 Section 1.2.3) 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).

### 1.2.2 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 Section 1.2.3).

- We contributed to the definition of E-Lotos (*Enhanced*-Lotos, Iso/Iec standard 15437:2001), a deep revision of Lotos, which tries to provide a greater expressiveness (for instance, by introducing quantitative time to describe systems with real-time constraints) together with a better user friendliness. Our contributions to E-LOTOS are available on the WEB (see "`http://www.inrialpes.fr/vasy/elotos`").

- We are also working on an E-LOTOS variant, named LOTOS NT (LOTOS *New Technology*) [GS98] [16], in which we can experiment new ideas more freely than in the constrained framework of an international standard. Like E-LOTOS, LOTOS NT consists of three parts: a *data part*, which allows the description of data types and functions, a *process part*, which extends the LOTOS process algebra with new constructs such as exceptions and quantitative time, and *modules*, which provide for structure and genericity. Both languages differ in that LOTOS NT combines imperative and functional features, and is also simpler than E-LOTOS in some respects (static typing, operator overloading, arrays), which should make it easier to implement. We are developing several tools for LOTOS NT: a prototype compiler named TRAIAN, a translator from (a subset of) LOTOS NT to LOTOS, and an intermediate semantic model named NTIF (*New Technology Intermediate Form*) [GL02].

### 1.2.3 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.

**The CADP Toolbox.** 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 [Gar89] 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 [GS90] is a compiler that translates LOTOS processes into either C code (for rapid prototyping and testing purposes) or finite graphs (for verification purpose).

[GS98] Hubert Garavel and Mihaela Sighireanu. Towards a Second Generation of Formal Description Techniques – Rationale for the Design of E-LOTOS. In Jan-Friso Groote, Bas Luttik, and Jos van Wamel, editors, *Proceedings of the 3rd International Workshop on Formal Methods for Industrial Critical Systems FMICS'98 (Amsterdam, The Netherlands)*, pages 187–230, Amsterdam, May 1998. CWI. Invited lecture.

[GL02] Hubert Garavel and Frédéric Lang. NTIF: A General Symbolic Model for Communicating Sequential Processes with Data. In Doron Peled and Moshe Vardi, editors, *Proceedings of the 22nd IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2002 (Houston, Texas, USA)*, volume 2529 of *Lecture Notes in Computer Science*, pages 276–291. Springer Verlag, November 2002. Full version available as INRIA Research Report RR-4666.

[Gar89] Hubert Garavel. Compilation of LOTOS Abstract Data Types. In Son T. Vuong, editor, *Proceedings of the 2nd International Conference on Formal Description Techniques FORTE'89 (Vancouver B.C., Canada)*, pages 147–162. North-Holland, December 1989.

[GS90] Hubert Garavel and Joseph Sifakis. Compilation and Verification of LOTOS Specifications. In L. Logrippo, R. L. Probert, and H. Ural, editors, *Proceedings of the 10th International Symposium on Protocol Specification, Testing and Verification (Ottawa, Canada)*, pages 379–394. IFIP, North-Holland, June 1990.

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 [Gar98] is a generic software environment for developing tools that explore graphs on the fly (for instance, simulation, verification, and test generation tools). Such tools can be developed independently from any particular high level language. In this respect, Open/Cæsar plays a central role in Cadp by connecting language-oriented tools with model-oriented tools. Open/Cæsar consists of a set of 16 code libraries with their programming interfaces, such as:

  - Caesar_Graph, which provides the programming interface for graph exploration,
  - Caesar_Hash, which contains several hash functions,
  - Caesar_Solve, which resolves boolean equation systems on the fly,
  - Caesar_Stack, which implements stacks for depth-first search exploration,
  - Caesar_Table, which handles tables of states, transitions, labels, etc.

  A number of tools have been developed within the Open/Cæsar environment, among which:

  - Bisimulator, which checks bisimulation equivalences and preorders,
  - Determinator, which eliminates nondeterminism in normal, probabilistic, or stochastic systems,
  - Distributor, which generates the graph of reachable states using several machines,
  - Evaluator, which evaluates regular alternation-free $\mu$-calculus formulas,
  - Executor, which performs random execution,
  - Exhibitor, which searches for execution sequences matching a given regular expression,
  - Generator, which constructs the graph of reachable states,
  - Projector, which computes abstractions of communicating systems,
  - Reductor, which constructs and minimizes the graph of reachable states modulo various equivalence relations,
  - Simulator, Xsimulator, and Ocis, which allow interactive simulation, and
  - Terminator, which searches for deadlock states.

- Bcg (*Binary Coded Graphs*) is both a file format for storing very large graphs on disk (using efficient compression techniques) and a software environment for handling this format. Bcg also plays a key role in Cadp as many tools rely on this format for their inputs/outputs. The Bcg environment consists of various libraries with their programming interfaces, and of several tools, such as:

[Gar98] Hubert Garavel. OPEN/CÆSAR: An Open Software Architecture for Verification, Simulation, and Testing. In Bernhard Steffen, editor, *Proceedings of the First International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'98 (Lisbon, Portugal)*, volume 1384 of *Lecture Notes in Computer Science*, pages 68–84, Berlin, March 1998. Springer Verlag. Full version available as INRIA Research Report RR-3352.

  - BCG_DRAW, which builds a two-dimensional view of a graph,
  - BCG_EDIT, which allows to modify interactively the graph layout produced by BCG_DRAW,
  - BCG_GRAPH, which generates various forms of practically useful graphs,
  - BCG_INFO, which displays various statistical information about a graph,
  - BCG_IO, which performs conversions between BCG and many other graph formats,
  - BCG_LABELS, which hides and/or renames (using regular expressions) the transition labels of a graph,
  - BCG_MERGE, which gathers graph fragments obtained from distributed graph construction,
  - BCG_MIN, which minimizes a graph modulo strong or branching equivalences (and can also deal with probabilistic and stochastic systems),
  - BCG_STEADY, which performs steady-state numerical analysis of (extended) continuous-time Markov chains,
  - BCG_TRANSIENT, which performs transient numerical analysis of (extended) continuous-time Markov chains, and
  - XTL (*eXecutable Temporal Language*), which is a high level, functional language for programming exploration algorithms on BCG graphs. XTL provides primitives to handle states, transitions, labels, *successor* and *predecessor* functions, etc. For instance, one can define recursive functions on sets of states, which allow to specify in XTL evaluation and diagnostic generation fixed point algorithms for usual temporal logics (such as HML [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.

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

[HM85] M. Hennessy and R. Milner. Algebraic Laws for Nondeterminism and Concurrency. *Journal of the ACM*, 32:137–161, 1985.

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, April 1986.

[NV90] R. De Nicola and F. W. Vaandrager. *Action versus State Based Logics for Transition Systems.* In *Semantics of Concurrency*, volume 469 of *Lecture Notes in Computer Science*, pages 407–419. Springer Verlag, 1990.

The Cadp tools are well-integrated and can be accessed easily using either the Eucalyptus graphical interface or the Svl [GL01] 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.

**The TRAIAN Compiler.** We develop a compiler named Traian for translating descriptions written in the Lotos NT language into C programs, which will be used for simulation, rapid prototyping, verification, and testing.

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

Although this version of Traian is still incomplete (it does not handle Lotos NT processes), it already has useful applications in compiler construction [GLM02]. The recent compilers developed by the Vasy project team — namely Aal, Chp2Lotos, Evaluator 4.0, Exp.open 2.0, Fsp2Lotos, Lnt2Lotos, Ntif, and Svl — 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`").

### 1.2.4 Application domains

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

[GL01] Hubert Garavel and Frédéric Lang. SVL: a Scripting Language for Compositional Verification. In Myungchul Kim, Byoungmoon Chin, Sungwon Kang, and Danhyung Lee, editors, *Proceedings of the 21st IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2001 (Cheju Island, Korea)*, pages 377–392. IFIP, Kluwer Academic Publishers, August 2001. Full version available as INRIA Research Report RR-4223.

[GLM02] Hubert Garavel, Frédéric Lang, and Radu Mateescu. Compiler Construction using LOTOS NT. In Nigel Horspool, editor, *Proceedings of the 11th International Conference on Compiler Construction CC 2002 (Grenoble, France)*, volume 2304 of *Lecture Notes in Computer Science*, pages 9–13. Springer Verlag, April 2002.

- *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.

## 1.3 Objectives for the evaluation period

At the previous project-team evaluation, we proposed to focus our activity during 2004-2007 on the following research themes:

1. **New generation formal description techniques.** The adequate modeling of asynchronous systems is an open problem, for which no full solution exists so far. In this domain, our previous work on the Lotos language is almost complete (except for some improvements asked and financed by Bull). We wish to focus on the new E-Lotos standard.

   The industrial partners to which we have presented E-Lotos have expressed their interest, but are still expecting tools. This is the reason why, based on the experience acquired with Lotos, we wish to design efficient compilation and verification techniques for E-Lotos. This work has started with the development of the Traian compiler and the definition of the Ntif intermediate model.

2. **Fight against state explosion.** We wish to continue our work to fight against the combinatorial explosion of the number of states, which happens in enumerative verification. We consider in particular three directions:

   - Symbolic analysis techniques, which consist in analysing the data-flow of functions and processes.
   - Compositional verification, which consists in processing separately (as much as possible) the sequential components of a parallel system under verification. The efficiency of this approach can be improved by combining it with techniques based on partial orders and the detection of symmetries. We have started a complete rebuild of the Cadp compositional verification tools, in order to make compositional verification more efficient and more automated.
   - Massively parallel verification, which aims at exploiting the resources (memory as well as Cpu time) provided by the parallel and distributed architectures, in particular clusters of Pcs. Our objective is to enable the scaling up of the Cadp verification tools in order to verify systems with at least one billion of "real" states (in explicit enumeration).

3. **Temporal logic extended with data.** Usual temporal logics permit to specify system correctness properties, but suffer from an important limitation. To palliate this limitation, we have extended the $\mu$-calculus with constructs permitting to reference the typed data contained in messages exchanged between distributed agents. Our partner Bull is looking forward to using this formalism. The next step consists in compiling and verifying properties of the $\mu$-calculus containing data efficiently.

4. **Generic components for verification, test, and performance evaluation.** For long, we have been delivering our research results in the form of software components, which permit to build tools of increasing sophistication. After the "classical" software components Bcg and Open/Cæsar (for the representation of explicit or implicit transition systems), we have recently designed new libraries for efficient on the fly resolution of boolean equation systems, which are likely to serve in verification based on the $\mu$-calculus and bisimulations (and perhaps also in test generation, partial order verification, and controller synthesis). We also work at other software tools that bring the domains of formal verification and performance evaluation nearer to each other.

During the evaluation period, the scope of Objective 1 was broadened by considering source-level translations from various process algebras and concurrent languages into LOTOS, so as to widen the applicability of the CADP tools. We undertook these new activities in the framework of several projects (FIACRE, OPENEMBEDD, TOPCASED, and the INRIA/LETI collaboration).

To assess the usability of LOTOS NT in an industrial environment, we also undertook the development of a prototype translator from LOTOS NT to LOTOS, which allows to benefit from all optimizations of the LOTOS compilers without investing too much resources in an optimized native implementation of LOTOS NT.

## 1.4 Objective 1: New generation formal description techniques

### 1.4.1 Personnel

Hubert Garavel, Frédéric Lang, Wendelin Serwe.

### 1.4.2 Project-team positioning

Vasy is the only Inria research team whose main research field is model checking of asynchronous systems (as opposed to model checking of synchronous systems, or other verification techniques). As formal description techniques, Vasy relies on process algebras, in particular the Iso standards Lotos and E-Lotos. As regards the definition of new formal description techniques, Vasy participated actively to the standardization of E-Lotos and cooperates with several European research groups (Cwi, Laas-Cnrs, Saarland University, ...). Vasy is the only team to develop and maintain tools for (a variant of) E-Lotos.

Vasy is also involved in two major French projects in the field of the integration of formal methods in model driven engineering, namely OpenEmbeDD and TopCased.

### 1.4.3 Scientific achievements

**Compilation of LOTOS.** The Cadp toolbox contains several tools dedicated to the Lotos language, namely: the Cæsar.adt compiler [Gar89] for the data type part of Lotos, the Cæsar compiler [GS90] for the process part of Lotos, and the Cæsar.indent pretty-printer.

As regards the Cæsar.adt compiler for the data part of Lotos:

- We designed and implemented a fixpoint algorithm for the detection of Lotos types whose domain of values is either finite or manually bounded by the specifier.

- 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 allow iterators to be generated for all finite Lotos types, including the intricate case of union types, which might be nested at any depth.

[Gar89] Hubert Garavel. Compilation of LOTOS Abstract Data Types. In Son T. Vuong, editor, *Proceedings of the 2nd International Conference on Formal Description Techniques FORTE'89 (Vancouver B.C., Canada)*, pages 147–162. North-Holland, December 1989.

[GS90] Hubert Garavel and Joseph Sifakis. Compilation and Verification of LOTOS Specifications. In L. Logrippo, R. L. Probert, and H. Ural, editors, *Proceedings of the 10th International Symposium on Protocol Specification, Testing and Verification (Ottawa, Canada)*, pages 379–394. IFIP, North-Holland, June 1990.

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.

As regards the Cæsar compiler for the process part of Lotos:

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

- In the framework of the FormalFame Plus contract, we simplified the use of the Exec/Cæsar environment [GVZ01]. Exec/Cæsar allows to interconnect, on the one hand, the C code generated by Cæsar from the Lotos description of a system and, on the other hand, the "real" environment with which this system interacts. This interconnection is implemented as a collection of C functions, one per visible gate declared in the Lotos specification, which have to be written by hand.

  The new version of Cæsar greatly automates this task by generating automatically, for each function, a C code skeleton that implements appropriate pattern-matching actions for checking gate parameters — since, in Lotos, the same gate can be overloaded with several parameter lists that differ in number, types and direction (input or output) — as well as logging actions to trace the execution of these functions.

- We 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 (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 worked on 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. We identified a difficult problem arising in the particular case of "*reset/use*" conflicts, and we refined our approach to handle such conflicts properly.

[GVZ01] Hubert Garavel, César Viho, and Massimo 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):314–331, July 2001. Also available as INRIA Research Report RR-4041.

[Gal93] Juan Galvez Londono. *Analyse du flux des données dans un système parallèle.* DEA, Institut National Polytechnique de Grenoble, June 1993.

We implemented our ideas in version 7.0 of CÆSAR (about 5,000 lines of additional C code), which was officially released as part of CADP in July 2006. On all CADP demos, CÆSAR 7.0 reduced the state space by a mean factor of 45 (we observed a maximum factor of 4,400) as regards the number of states and by a mean factor of 38 (we observed a maximum factor of 3,100) as regards the number of transitions. On 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 $10^4$.

This work led to publications [19, 4].

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

We performed maintenance activities for these tools (1 bug fixed in CÆSAR.ADT, 5 bugs fixed in CÆSAR, 1 bug fixed in the CÆSAR.BDD tool invoked by CÆSAR, 3 bug fixes and 2 enhancements in CÆSAR.INDENT, and 4 bugs fixed in the common front-end). We also improved the C code generated by CÆSAR and CÆSAR.ADT to avoid warnings emitted by the most recent C compilers.

**Compilation of E-LOTOS and LOTOS NT.** As regards the data part of E-LOTOS—and, more specifically, its LOTOS NT variant elaborated by the VASY team — we continued to improve the TRAIAN compiler, which is distributed on the Internet and used intensively within the VASY team as a development tool for compiler construction [GLM02].

During the evaluation period, we released three successive versions 2.3, 2.4, and 2.5 of TRAIAN. This development effort, which increased the software size from 48,000 to 55,000 lines of code, completes the integration in TRAIAN of the code optimizations studied by Claude Chaudet in 1999 (see § 5.2.3 in the 1999 VASY activity report and § 5.2.1 in the 2002 VASY activity report). It also brings a higher degree of symmetry between TRAIAN and the CÆSAR.ADT compiler for the data part of LOTOS. In addition to several bug fixes, the new version of TRAIAN brings useful enhancements:

- Particular classes of LOTOS NT data types (enumerated types, tuples, natural numbers, singleton types, and isomorphic types) are now recognized automatically and implemented optimally.
- For recursive types, heuristics allow to reduce the number of LOTOS NT types implemented using pointers; however, a compiler directive exists to force a given LOTOS NT type to be implemented using pointers.

[GLM02] Hubert Garavel, Frédéric Lang, and Radu Mateescu. Compiler Construction using LOTOS NT. In Nigel Horspool, editor, *Proceedings of the 11th International Conference on Compiler Construction CC 2002 (Grenoble, France)*, volume 2304 of *Lecture Notes in Computer Science*, pages 9–13. Springer Verlag, April 2002.

- It is now possible to give pointer types a canonical representation by storing all their values into hash tables, which avoids to allocate multiple instances of the same value; in the framework of enumerative verification, this technique allows significant savings in memory space (on a benchmark proposed by Jan Friso Groote, we observed that the amount of memory needed was divided by 400).

- As for LOTOS, it is now possible to split LOTOS NT specifications into several files using a compiler directive.

- The TRAIAN compiler was ported to the MAC OS X and recent LINUX operating systems, and the C code generated by TRAIAN was made compatible with the latest versions of GCC and Intel's ICC compilers. In addition, the TRAIAN libraries and shell-scripts have been ported to the ITANIUM 64-bit platform running the LINUX operating system.

We also developed a new tool, named TRAIAN.INDENT, for indenting LOTOS NT programs, similar to the existing tool CÆSAR.INDENT.

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

In the framework of the FORMALFAME PLUS contract, we undertook the development of a tool suite for the translation from LOTOS NT to LOTOS, which aims at easing the development of large specifications by BULL and to reuse the existing LOTOS tools for analyzing concurrent systems described in LOTOS NT. The tool suite consists of a LOTOS/LOTOS NT preprocessing tool named LPP (1,280 lines of C code), a tranlator from LOTOS NT data types and functions to LOTOS named LNT2LOTOS, developed using the aforementioned SYNTAX/TRAIAN technology (2,100 lines of SYNTAX code, 14,000 lines of LOTOS NT code, and 1,200 lines of C code), and a shell script named LNT_COMPILE, which calls LPP and LNT2LOTOS and allows the generated LOTOS code to be combined with handwritten LOTOS code and/or C code provided by the user. The LNT2LOTOS translator has the following features:

- It translates all LOTOS NT data types into LOTOS abstract data types. All constructor types are equiped with predefined comparison functions ("=", "<", "≤", etc.) based on a recursive lexicographic ordering of constructors and constructor fields, if not given by the user otherwise. A new predefined type "**sorted list of** *T*" was added to the LOTOS NT language. The elements of a sorted list are sorted automatically using the order relation "<" on *T*.

- It translates LOTOS NT functions into LOTOS equations. The translation was inspired from an algorithm [PFK05] that translates into Horn clauses a subset of the C language ("**while**" loops without "**break**" statements and functions with value passing parameters only and a single "**return**" statement located at the end of the function). We extended this algorithm so as to handle reference passing parameters, pattern matching ("**case**" statement), loop interruptions ("**break**" statement), multiple "**return**" statements within the body of functions, uncatchable exceptions ("**raise**" statement), and function name overloading.

[PFK05] Olivier Ponsini, Caroline Fédèle, and Emmanuel Kounalis. Rewriting of imperative programs into logical equations. *Science of Computer Programming*, 56(3):363–401, May – June 2005.

7 successive versions of the tool suite were delivered to Bull, who uses Lotos NT to model a critical part of its Fame2 multiprocessor architecture for high-end servers. A non-regression test suite of 67 programs representing more than 6,000 lines of Lotos NT code was developed. A 47 page reference manual was written [39]. A forge was set up under Inria GForge to track bugs and feature requests, and to serve as a repository where our Bull partners can download the new versions of the Lnt2Lotos tool suite.

As regards the process part of E-Lotos and Lotos NT, compiling 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 [GL02].

The existing tools for Ntif were enhanced in several ways:

- For modularity reasons, we merged the Nt2Dot (which visualizes Ntif descriptions graphically) and Nt2If (which unfolds Ntif descriptions to produce lower level formalisms) into one single tool, named Ntif. The architecture of this new tool supports several compiler back-ends that translate Ntif into a variety of languages and formats.

- A file inclusion mechanism was implemented, which allows to split Ntif descriptions into several files.

- The static semantics checking phase of Ntif was entirely rewritten to be more efficient and display better error messages. Static checks for proper variable initialization were added, which allowed to detect uninitialized variables in existing Ntif specifications.

- Two new back-ends were developed, which translate Ntif to the input languages used by the TReX and Uppaal tools for timed automata. In the case of Uppaal, both Xta and Xml formats can be generated. The back-ends translate the high level Ntif timed constructs into clocks, time guards, and time progress conditions that express the impossibility to enter or stay in a given state.

- We added to Ntif two standard libraries (lossy buffers and write-only buffers). In the general case, these buffers are expressed as normal Ntif processes. However, when translating to TReX input language, these buffers are recognized and implemented as TReX built-in buffers for optimization purpose.

[GL02] Hubert Garavel and Frédéric Lang. NTIF: A General Symbolic Model for Communicating Sequential Processes with Data. In Doron Peled and Moshe Vardi, editors, *Proceedings of the 22nd IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2002 (Houston, Texas, USA)*, volume 2529 of *Lecture Notes in Computer Science*, pages 276–291. Springer Verlag, November 2002. Full version available as INRIA Research Report RR-4666.

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

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

**Source-Level Translations between Concurrent Languages.** To enhance the dissemination of process algebras in industry and to widen the applicability of the CADP tools, we investigated source-level translations from various concurrent languages to LOTOS:

- In the framework of the INRIA/LETI collaboration, we focused on the process algebra CHP (*Communicating Hardware Processes*), which is used by the LETI laboratory to describe complex, asynchronous circuits at a high abstraction level. Having the goal to integrate formal verification into the design flow of complex microelectronic circuits, we defined a structural operational semantics for CHP and we proposed a translation scheme from CHP to LOTOS, which is finely-tuned for handling the hardware-specific "probe" operator of CHP. We developed a translator named CHP2LOTOS [36] (2,200 lines of SYNTAX code, 13,400 lines of LOTOS NT code, and 3,900 lines of C code), which was successfully applied for verifying an asynchronous circuit implementing the DES encryption standard, as well as the FAUST asynchronous NOC (*Network on Chip*) [SSTV07] developed at LETI.

- We considered the process algebra FSP (*Finite State Processes*) defined in a popular textbook on concurrency [MK99]. In collaboration with Jeff Kramer and Jeff Magee (Imperial College, London), we designed a translation scheme from FSP to LOTOS. This allowed to detect and remove several ambiguities in the reference FSP grammar. We developed a prototype translator FSP2LOTOS (5,000 lines of SYNTAX code, 20,000 lines of LOTOS NT code, and 500 lines of C code) and successfully tested it on 10,500 lines of FSP code, including many examples given in the FSP textbook.

- In the framework of the FIACRE, OPENEMBEDD, and TOPCASED projects, and in cooperation with the LAAS-CNRS and IRIT laboratories, we defined a new intermediate model named FIACRE (*Format Intermédiaire pour les Architectures de Composants Répartis Embarqués*). Derived from NTIF [GL02] and V-COTRE [BRV+03],

[SSTV07] Gwen Salaün, Wendelin Serwe, Yvain Thonnart, and Pascal Vivet. Formal Verification of CHP Specifications with CADP — Illustration on an Asynchronous Network-on-Chip. In *Proceedings of the 13th IEEE International Symposium on Asynchronous Circuits and Systems ASYNC 2007 (Berkeley, California, USA)*. IEEE Computer Society Press, 2007. to appear.

[MK99] Jeff Magee and Jeff Kramer. *Concurrency: State Models and Java Programs*. Wiley, 1999.

[GL02] Hubert Garavel and Frédéric Lang. NTIF: A General Symbolic Model for Communicating Sequential Processes with Data. In Doron Peled and Moshe Vardi, editors, *Proceedings of the 22nd IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2002 (Houston, Texas, USA)*, volume 2529 of *Lecture Notes in Computer Science*, pages 276–291. Springer Verlag, November 2002. Full version available as INRIA Research Report RR-4666.

[BRV+03] B. Berthomieu, P.O. Ribet, F. Vernadat, J. Bernartt, J.-M. Farines, J.-P. Bodeveix, M. Filali, G. Padiou, P. Michel, P. Farail, P. Gaufillet, P. Dissaux, and J.-L. Lambert. Towards the verification of real-time systems in avionics: the COTRE approach. In Thomas Arts and Wan Fokkink, editors, *Proceedings of the 8th International Workshop on Formal Methods for Industrial Critical Systems FMICS'2003 (Trondheim, Norway)*, volume 80 of *Electronic Notes on Theoretical Computer Science*, pages 201–216. Elsevier, June 2003.

Fiacre will be used as a pivot formalism between modeling languages (such as Aadl, Uml, or SysML) and verification tools (such as Cadp and Tina).

- In collaboration with Gregor Goessler (PopArt project-team), we designed a prototype translator from the Bip modeling language of the Prometheus tool developed by PopArt to the input languages of Cadp (Exp.open 2.0 and Svl).

- We also started to investigate the verification of Tlm (*Transaction Level Model*) specifications described in SystemC [IEE05]. For this purpose, we studied the SystemC front-end Pinapa [MMMC05], with the goal of devising a translation from (a Tlm subset of) SystemC into Lotos or Lotos NT.

### 1.4.4 Collaborations

- Gregor Goessler (PopArt project team) : connection to Cadp of the Prometheus tool's input language.

- Jeff Magee and Jeff Kramer (Imperial College, London) : translation from Fsp to Lotos; one month visit by Gwen Salaün at Imperial College.

- Yvain Thonnart and Pascal Vivet (Cea/Leti, Grenoble) : translation from Chp to Lotos; accepted joint publication [37].

- François Vernadat and Bernard Berthomieu (Laas-Cnrs, Toulouse) : design of Fiacre.

### 1.4.5 External support

The contracts Fiacre, FormalFame, FormalFame Plus, and Senva provided some funding to tackle this objective.

### 1.4.6 Self assessment

There has been a strong demand from Bull for new generation formal description techniques, which are the main topic of the FormalFame Plus collaboration. The results obtained so far are encouraging: Recently, Bull has started to use the data part of the Lotos NT language and gave us very positive feedback. Other industrial partners, such as ST Microelectronics, have also shown a strong interest in Lotos NT. This work should be extended to a subset of Lotos NT including processes. The translation to Lotos became a viable solution due to the major enhancements brought to Lotos compilers of Vasy (iterators, static analysis, ...).

There is also a strong interest from industrials for the connection of formal verification tools to model driven engineering environments. This is a topic of projects such as TopCased and OpenEmbeDD, which involve numerous industrial partners such as Airbus, and to which Vasy participates. Such connections necessitate carefully designed intermediate models between modeling languages and verification tools. Our work on Ntif and Fiacre is an important step in that direction.

---

[IEE05] IEEE. IEEE Standard SystemC Language Reference Manual. IEEE Standard 1666-2005, Institution of Electrical and Electronic Engineers, December 2005.

[MMMC05] Matthieu Moy, Florence Maraninchi, and Laurent Maillet-Contoz. Pinapa: An Extraction Tool for SystemC descriptions of Systems-on-a-Chip. In Wayne Wolf, editor, *EMSOFT*. ACM, September 2005.

## 1.5 Objective 2: Fight against state explosion

### 1.5.1 Personnel

Hubert Garavel, Christophe Joubert, Frédéric Lang, Radu Mateescu, Wendelin Serwe.

### 1.5.2 Project-team positioning

Any research team tackling enumerative verification of asynchronous concurrent systems is faced to state explosion. Therefore, numerous techniques to alleviate the state explosion problem have been proposed. VASY focuses on the smooth integration and combination of most of these techniques (symbolic analysis, compositional verification, on the fly verification, massively parallel verification, ... ) in a single toolbox. As regards compositional verification, VASY innovated by the development of a scripting language and compiler which eases the writing of compositional verification scenarios. VASY is also one of (if not the) first teams to have made available a tool for massively parallel verification. VASY collaborates with the leading European teams in massively parallel verification within the EC-MOAN project.

### 1.5.3 Scientific achievements

**Symbolic analysis techniques.** We developed symbolic analysis techniques for state space reduction based on live variable analysis within CÆSAR. See Section 1.4.3.

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

Designed to speed up the compositional verification of asynchronous systems, BCG_GRAPH is a tool for generating the BCG graphs corresponding to FIFO communication buffers efficiently. During the evaluation period, we extended BCG_GRAPH so as to generate two other kinds of graphs, namely *bag automata*, which model asynchronous communication buffers that do not preserve the ordering of messages, and *chaos automata*, which are graphs containing one single state and and a set of looping transitions on that state. BCG_GRAPH (2,700 lines of C code) allows to generate large BCG graphs (hundreds of thousands of states) within a few minutes. The generated graphs are always minimal modulo strong bisimulation.

PROJECTOR 2.0 is a tool (totally rewritten in 2002) that implements behaviour abstraction [GSL96,KM97], by taking into account interface constraints. During the evaluation period, we improved PROJECTOR 2.0 by adding options to hide and rename labels on the fly, based on the CÆSAR_SOLVE library, and we corrected a few bugs. We also improved its efficiency by introducing a hash function specifically adapted to state products. On real examples provided by the Technical University of Eindhoven, the execution time of PROJECTOR 2.0 was divided by a factor of up to four. A manual page was written for PROJECTOR 2.0 and the tool became part of CADP in December 2004.

[GSL96] S. Graf, B. Steffen, and G. Lüttgen. Compositional Minimization of Finite State Systems using Interface Specifications. *Formal Aspects of Computation*, 8(5):607–616, September 1996.

[KM97] Jean-Pierre Krimm and Laurent Mounier. Compositional State Space Generation from LOTOS Programs. In Ed Brinksma, editor, *Proceedings of TACAS'97 Tools and Algorithms for the Construction and Analysis of Systems (University of Twente, Enschede, The Netherlands)*, volume 1217 of *Lecture Notes in Computer Science*, Berlin, April 1997. Springer Verlag. Extended version with proofs available as Research Report VERIMAG RR97-01.

EXP.OPEN 2.0 is a compositional verification tool that explores on the fly the graph corresponding to a network of communicating automata (represented as a set of BCG files). These automata are composed together in parallel using either algebraic operators (as in CCS, CSP, LOTOS, and $\mu$CRL), "graphical" operators (as in E-LOTOS [ISO01] and LOTOS NT [GS99]), or synchronization vectors (as in the MEC and FC2 tools). Additional operators are available to hide and/or rename labels (using regular expressions) and to cut certain transitions.

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

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

- We took $\mu$CRL syntax conventions into account to extract information (gate, offers) from labels, thus improving $\mu$CRL 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 added options to obtain static information about the network of communicating automata, such as a list of the labels that potentially belong to the product and the size of each BCG graph in the network.

- We added a new operator for specifying priorities between the transitions of a network of communicating automata.

- The code was optimized and the algorithm for enumerating the successors of a given state was improved, which reduced the generation time by 20 % on average, with a constant negligible memory overhead. A performance comparison with EXP.OPEN 1.0 on a dozen realistic examples shows that EXP.OPEN 2.0 is faster (from 1.6 to 50 times) and uses less memory (about 2 times less).

- We extended the possibilities of translation from EXP.OPEN's networks of communicating automata into input formats accepted by other tools. In addition to the existing interconnection with the PEP tool for Petri nets (developed at the University of Oldenburg), we implemented translations to the parallel FC2 format designed at INRIA Sophia-Antipolis and to the TPN Petri nets format of the TINA toolbox developed at LAAS-CNRS.

- To address state explosion, we enhanced EXP.OPEN 2.0 with four partial order reduction techniques.

  The first and second techniques preserve respectively the deadlocks and the weak traces of the network of automata.

---

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

[GS99] Hubert Garavel and Mihaela Sighireanu. A Graphical Parallel Composition Operator for Process Algebras. In Jianping Wu, Qiang Gao, and Samuel T. Chanson, editors, *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)*, pages 185–202. IFIP, Kluwer Academic Publishers, October 1999.

The third technique is based on $\tau$-confluence and preserves branching bisimulation. We assessed our approach on two benchmarks (a leader election protocol on a token ring, and a distributed implementation of Erathostene's sieve) used by Ramakrishna & Smolka [RS97] to experiment another partial order reduction based on $\tau$-inertness. The graphs generated by Exp.open 2.0 are comparable or even smaller than those reported in [RS97]: As the number of concurrent processes increases, the graph generated by Exp.open 2.0 for the sieve remains of constant size, whereas [RS97] indicates a linear growth.

The fourth technique (based on observations made in [Her02]) applies to stochastic models and preserves stochastic branching bisimulation. It 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. This allowed to divide by up to 5 the number of states of the generated stochastic models.

- We also developed 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, 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.

The Exp.open 2.0 tool consists of 2,200 lines of Syntax code, 8,800 lines of Lotos NT code, and 2,400 lines of C code. A detailed manual page for Exp.open 2.0 was written and the tool became part of Cadp in August 2004. Exp.open 2.0 was used in the framework of the Fiacre national action and at Saarland University, among other places. We developed three new demo examples to illustrate the recent functionalities of Exp.open 2.0.

Svl (Script Verification Language) is both a high level language for expressing complex verification scenarios and a compiler dedicated to this language. During the evaluation period, we enhanced the Svl language and compiler along the following lines:

- We modified Svl so as to invoke the new Bisimulator tool to perform behavioural comparisons, Projector 2.0 to implement its abstraction operator, and Exp.open 2.0 to compute automata products.

- We extended the Svl language to support the new equivalences available in Reductor 5.0 and the new features of Bisimulator and Evaluator, e.g., selection between depth-first or breadth-first search algorithms, selection of algorithms dedicated to acyclic graphs, etc.

[RS97] Y.S. Ramakrishna and S.A. Smolka. Partial-Order Reduction in the Weak Modal Mu-Calculus. In A. Mazurkiewicz and J. Winkowski, editors, *Proceedings of the 8th International Conference on Concurrency Theory CONCUR'97*, volume 1243 of *Lecture Notes in Computer Science*, pages 5–24. Springer Verlag, 1997.

[Her02] Holger Hermanns. *Interactive Markov Chains and the Quest for Quantified Quality*, volume 2428 of *LNCS*. Springer Verlag, 2002.

- We added a new operator called "*refined abstraction*", which allows to generate the graph of a process under constraints generated automatically using Exp.open 2.0.
- We adapted Svl so that, depending on the equivalence to be preserved, it invokes Exp.open 2.0 with the most appropriate partial order reduction, which is inferred from the semantic context automatically.

Two articles, one on Exp.open 2.0 [26] and one on refined interface generation using Exp.open 2.0 and Svl [27] were published in international conferences.

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

As a first goal, we focused on parallelizing the graph construction algorithm, which is a bottleneck for verification, as it requires a considerable amount of memory to store all reachable states. For this purpose, we developed two tools [GMS01]: Distributor splits the construction of a graph over $N$ machines communicating using Tcp/Ip sockets; each machine builds a graph fragment, the distribution of states between the machines being determined by a static hash function; Bcg_Merge merges the $N$ graph fragments constructed by Distributor to produce the entire graph. Distributor 3.0 and Bcg_Merge 3.0 [18] were properly documented and became part of Cadp in 2005.

In the framework of the Senva collaboration, we defined in collaboration with Stefan Blom (Cwi, Amsterdam) the Pbg (*Partitioned* Bcg *Graphs*) format to represent the concept of "partitioned labeled transition system" [GMS01], which was implemented in Distributor and Bcg_Merge, allowing to simplify the command-line interface of these tools. The Pbg format provides various information for handling these fragments (number of states and transitions of each fragment, Gcf file used to generate the fragments, log files produced on each machine, etc.). We developed four new prototype tools operating on Pbg files: Pbg_Cp copies a Pbg file and its dependencies (fragments, log files, and Gcf file) from a machine to another; Pbg_Mv moves a Pbg file (and its dependencies) between two machines; Pbg_Rm removes a Pbg file (and its dependencies); and Pbg_Open provides a distributed algorithm that implements the Open/Cæsar programming interface [Gar98], thus allowing to explore on the fly a Pbg file (without merging its fragments first as Bcg_Merge does).

To support the communication underlying distributed tools, we designed a library called Cæsar_Network, which provides various functionalities: 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, error processing, etc. Distributed computing sessions are described using the Gcf (*Grid Configuration*

[GMS01] Hubert Garavel, Radu Mateescu, and Irina Smarandache. Parallel State Space Construction for Model-Checking. In Matthew B. Dwyer, editor, *Proceedings of the 8th International SPIN Workshop on Model Checking of Software SPIN'2001 (Toronto, Canada)*, volume 2057 of *Lecture Notes in Computer Science*, pages 217–234, Berlin, May 2001. Springer Verlag. Revised version available as INRIA Research Report RR-4341 (December 2001).

[Gar98] Hubert Garavel. OPEN/CÆSAR: An Open Software Architecture for Verification, Simulation, and Testing. In Bernhard Steffen, editor, *Proceedings of the First International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'98 (Lisbon, Portugal)*, volume 1384 of *Lecture Notes in Computer Science*, pages 68–84, Berlin, March 1998. Springer Verlag. Full version available as INRIA Research Report RR-3352.

*File*) format, which was recently extended in order to support the mainstream job schedulers available in clusters and grids. Cæsar_Network allows a clear separation between verification algorithms and communication primitives.

As a second goal, we aim at parallelizing on the fly verification itself. Because the Cæsar_Solve library is our central verification engine for both model checking, e.g., in the Evaluator tool, and equivalence checking, e.g., in the Bisimulator and Reductor tools, we undertook the development of a distributed version of the Cæsar_Solve library that could solve boolean equation systems on the fly using several machines. So far, we have developed a distributed local resolution algorithm [24, 2] based upon a breadth-first search of the underlying boolean graph performed by several worker processes under the supervision of a coordinator process. This distributed algorithm ($16,000$ lines of C code) also produces 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.). It was recently enhanced to detect on the fly cyclic dependencies between equation blocks (the presence of such dependencies indicates that the boolean equation system has an alternation depth greater than one).

In order to obtain distributed on the fly equivalence checking and model checking functionalities, we developed prototype connections of the Bisimulator and Evaluator 3.5 tools, and of Reductor's $\tau$-confluence reduction, to the distributed boolean resolution algorithm. We also developed a random generator of boolean equation systems in order to test the performance of the distributed algorithm. An extensive set of experiments performed with these tools 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, low memory overhead, and a balanced distribution of work among machines [23, 12].

Finally, along the lines of the test generation theory [JJ05] implemented in the Tgv tool of Cadp, we developed a prototype tool named Extractor ($2,500$ lines of C code), which reformulates the test generation problem as the local resolution of a boolean equation system, performed using either the sequential algorithms of Cæsar_Solve, or the distributed resolution algorithm. The experiments performed on various graphs shown that the sequential version of Extractor exhibits performances comparable with Tgv, and confirmed the good behaviour of the distributed resolution algorithm [25].

### 1.5.4 Collaborations

- Stefan Blom, Bert Lisser, and Jaco van de Pol (Cwi, Amsterdam, The Netherlands) : definition of the Pbg format for representing partitioned transition systems.

- Holger Hermanns and Sven Johr (Saarland University, Germany) : stochastic partial order reduction in Exp.open 2.0.

[JJ05] Claude Jard and Thierry Jéron. TGV: Theory, Principles and Algorithms — A Tool for the Automatic Synthesis of Conformance Test Cases for Non-Deterministic Reactive Systems. *Springer International Journal on Software Tools for Technology Transfer (STTT)*, 7(4):297–315, August 2005.

### 1.5.5 External support

The contracts Fiacre, FormalFame, FormalFame Plus, Multival, OpenEmbeDD, Parfums, Senva, and Topcased provided funding to tackle this objective.

### 1.5.6 Self assessment

During the evaluation period, we implemented symbolic analysis techniques, which reduced the state spaces up to several orders of magnitude, while preserving strong bisimulation. We have started a study of other symbolic techniques, which have a strong potential impact on the efficiency of model checking.

During the evaluation period, our work in compositional verification continued with the aim to make compositional verification more practical, for instance by generating automatically efficient and correct interfaces. We integrated within the Svl scripting language the most recent functionalities provided by the Cadp tools. However, although facilitated by Svl, the choice of a successful compositional verification strategy remains a difficult task. Research should be continued in this direction.

The research on parallel and distributed verification currently receives a considerable attention from the research community. Although the distributed algorithms and tools available (or under development) in Cadp exhibit good performance on clusters of machines, it will be necessary to extend them in order to exploit the computing power of heterogeneous platforms such as grids.

## 1.6 Objective 3: Temporal logic extended with data

### 1.6.1 Personnel

Hubert Garavel, Radu Mateescu.

### 1.6.2 Project-team positioning

The model checking of the modal $\mu$-calculus on finite transition systems is a long-standing problem, which has received a considerable attention from the scientific community during the last two decades. However, despite the fact that numerous algorithms were proposed in the literature [EL86,AC88,CS93,VL92,And94,LRS98,DSC99], relatively few of them led to efficient and robust implementations. Among the tool environments offering $\mu$-calculus model checking features, the most prominent are the Concurrency Factory [CLSS96] and the Edinburgh Concurrency Workbench [SS98].

By extending the alternation-free modal $\mu$-calculus with ACTL-like action formulas and PDL-like modalities, we aimed at improving the conciseness and readability of formulas, whilst keeping a linear-time model checking complexity. As far as we know, the EVALUATOR 3.5 model checker of CADP is one of the few currently available tools providing this kind of extensions. The tool was successfully applied to 28 industrial case-studies (see "`http://www.inrialpes.fr/vasy/case-studies`") and is used as verification engine for analyzing video-on-demand server architectures [Pen06].

As regards the extensions of modal $\mu$-calculus with data-handling constructs, the existing tools are focused on different approaches, such as Prolog XSB resolution [LRS98] or

[EL86] E. A. Emerson and C-L. Lei. Efficient Model Checking in Fragments of the Propositional Mu-Calculus. In *Proceedings of the 1st LICS*, pages 267–278, 1986.

[AC88] A. Arnold and P. Crubillé. A linear algorithm to solve fixed-point equations on transition systems. *Information Processing Letters*, 29:57–66, 1988.

[CS93] Rance Cleaveland and Bernhard Steffen. A Linear-Time Model-Checking Algorithm for the Alternation-Free Modal Mu-Calculus. *Formal Methods in System Design*, 2(2):121–147, April 1993.

[VL92] B. Vergauwen and J. Lewi. A linear algorithm for solving fixed-point equations on transition systems. In *Proceedings of the 17th Colloquium on Trees in Algebra and Programming CAAP '92 (Rennes, France)*, volume 581 of *Lecture Notes in Computer Science*, pages 322–341, Berlin, February 1992. Springer Verlag.

[And94] H. R. Andersen. Model checking and boolean graphs. *Theoretical Computer Science*, 126(1):3–30, April 1994.

[LRS98] X. Liu, C. R. Ramakrishnan, and S. A. Smolka. Fully Local and Efficient Evaluation of Alternating Fixed Points. In Bernhard Steffen, editor, *Proceedings of 1st International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'98 (Lisbon, Portugal)*, volume 1384 of *LNCS*, pages 5–19. Springer Verlag, March 1998.

[DSC99] Xiaoqun Du, Scott A. Smolka, and Rance Cleaveland. Local Model Checking and Protocol Analysis. *Springer International Journal on Software Tools for Technology Transfer (STTT)*, 2(3):219–241, 1999.

[CLSS96] Rance Cleaveland, Philip M. Lewis, Scott A. Smolka, and Oleg Sokolsky. The Concurrency Factory: A Development Environment for Concurrent Systems. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th International Conference on Computer Aided Verification CAV'96 (New Brunswick, New Jersey, USA)*, volume 1102 of *Lecture Notes in Computer Science*, pages 398–401. Springer Verlag, August 1996.

[SS98] Perdita Stevens and Colin Stirling. Practical Model-Checking using Games. In Bernhard Steffen, editor, *Proceedings of the First International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'98 (Lisbon, Portugal)*, volume 1384 of *Lecture Notes in Computer Science*, pages 85–101, Berlin, March 1998. Springer Verlag.

[Pen06] Juan José Sánchez Penas. *From Software Architecture to Formal Verification of a Distributed System.* PhD thesis, University of Corunha, November 2006.

symbolic verification [Dam94]. Our Mcl language subsumes these proposals and enhances them with constructs inspired from programming languages and from hybrid logics. The Mcl model checking approach we implemented in the Evaluator 4.0 prototype tool, based on the local resolution of boolean equation systems, allows to reuse the efficient algorithms available in the Cæsar_Solve library of Cadp.

### 1.6.3 Scientific achievements

**The EVALUATOR Tool.** Evaluator is a model checker that evaluates a temporal logic property on a graph represented implicitly using the Open/Cæsar environment. Properties are described in regular alternation-free $\mu$-calculus [9], a logic built from boolean operators, modalities containing regular expressions denoting transition sequences, and fixed point operators without mutual recursion between least and greatest fixed points. Evaluator works on the fly, by reformulating the model checking problem as the local resolution of a boolean equation system. The tool generates diagnostics (examples and counterexamples) explaining why a formula is true or false, and also allows to define parameterized temporal operators and to group them into separate libraries.

The version Evaluator 3.5 (5, 600 lines of Syntax/Fnc2 code and 5, 100 lines of C code) became part of Cadp in 2005. This version uses the boolean resolution algorithms provided by the Cæsar_Solve library, whereas Evaluator 3.0 contained an ad hoc resolution engine. This enhances both the modularity and the functionality of the tool: the breadth-first search based algorithm A2 produces small-depth diagnostics; the memory-efficient algorithm A3 checks properties on acyclic graphs; and the memory-efficient algorithm A4 evaluates formulas of Ctl, Actl, and Pdl (which lead to disjunctive or conjunctive boolean equation systems) by storing only states of the graph. A prototype connection of Evaluator 3.5 to a distributed boolean resolution algorithm was developed and tested on graphs from the Vlts benchmark suite [25].

We also worked on extending the regular alternation-free $\mu$-calculus with new operators dedicated to the specification of properties involving data values. This led to the definition of Mcl (*Model Checking Language*), a language which extends the regular alternation-free $\mu$-calculus with the following constructs: data-handling operators inspired from programming languages, such as "**if-then-else**" and "**case**"; fixed point operators parameterized by data variables; action patterns extracting the values present on transition labels; regular expressions over transition sequences equipped with iteration operators ranging over natural intervals, and also with programming language constructs such as "**while**", "**until**", and "**for**"; and special operators for capturing states of the graph and manipulating them in formulas, which allows to express non-standard properties (e.g., the existence of self-loops) and past-time properties. We implemented the translation of Mcl formulas into boolean equation systems in a prototype tool Evaluator 4.0 (37, 600 lines of Syntax/Lotos NT code and 11, 100 lines of C code), which was successfully tested on 2, 300 examples of Mcl formulas and on all regular alternation-free $\mu$-calculus formulas available in the demo examples of Cadp.

**The AAL Tool.** In the framework of the ArchWare European project, we focused on the analysis of software architectures, by contributing to the definition of Aal (*Architecture Analysis Language*) [44], a language for expressing properties of software architectures and architectural styles. Aal contains operators borrowed from first-order logic and modal $\mu$-calculus, extended with predicates specific to architectural descriptions. It

[Dam94] M. Dam. Model Checking Mobile Processes (Full version). Research Report RR 94:1, Swedish Institute of Computer Science, Kista, Sweden, 1994.

allows to specify both style-related structural properties (e.g., connectivity between components, cardinality, etc.) and architecture-related behavioral properties (e.g., safety, liveness, fairness). We also defined AAF-MC (*Architecture Analysis Formalism for Model Checking*) [47], the fragment of AAL containing properties to be verified using model checking. For this fragment, we developed a prototype tool [48] ($7,500$ lines of code), which translates the temporal formulas expressed in AAF-MC into boolean equation systems. Finally, we developed a methodology for the efficient verification of AAF-MC properties on execution traces generated during the simulation of an architectural description [50].

### 1.6.4 Collaborations

We collaborated with BULL in the framework of the FORMALFAME and FORMALFAME PLUS contracts. David Champelovier enhanced the formula parameterization mechanisms in the input language of EVALUATOR 3.5. BULL engineers also provided valuable feedback and suggested various improvements of the tool.

### 1.6.5 External support

The contracts ARCHWARE, FORMALFAME, and FORMALFAME PLUS provided support to tackle this objective.

### 1.6.6 Self assessment

The on the fly model checking of modal $\mu$-calculi extended with data, such as the MCL language, is a difficult topic for which very rare model checkers are currently available. Therefore, we consider the proper integration of the EVALUATOR 4.0 model checker into CADP as an important activity, of which relatively few aspects remain to be carried out (testing, debugging, documentation). We expect that the tool will be most useful to our partners in the MULTIVAL project (BULL, ST MICROELECTRONICS, CEA/LETI) for analyzing multiprocessor architectures and asynchronous circuits, which involve significant data manipulation.

Among the possible enhancements of EVALUATOR 4.0 is a tighter connection of the tool with the data types and functions described in the LOTOS specifications under analysis: ideally, the user should be able to reference freely all these types and functions in the MCL formulas. This enhancement would require an appropriate extension of the OPEN/CÆSAR graph exploration interface.

## 1.7 Objective 4: Generic components for verification, test, and performance evaluation

### 1.7.1 Personnel

Hubert Garavel, Christophe Joubert, Frédéric Lang, Radu Mateescu, Wendelin Serwe.

### 1.7.2 Project-team positioning

Verification tools are complex software artifacts, whose development, optimization, and maintenance require significant effort and time. Moreover, these tools must exhibit high quality and robustness, since they are used in practice for assessing the quality of other software systems. For this purpose, we seek to promote modular architectures for verification tools, which reduce the development effort, allow the independent optimization of different functionalities, and improve the robustness of components by allowing their usage in different contexts.

In the context of CADP, a first step towards this direction was made by the OPEN/CÆSAR environment [Gar98], which defined a generic interface for representing transition systems, thus allowing to separate the aspects related to the compilation of a system description from the aspects related to on the fly verification. It is worth noticing that other recent verification tools have followed this track by using a modular architecture also, e.g., the BOGOR model checker for JAVA programs and the NUSMV symbolic model checker.

We continued the development of OPEN/CÆSAR by adding new generic libraries, such as the CÆSAR_SOLVE library for local resolution of alternation-free boolean equation systems, which can represent several verification problems (model checking, equivalence checking, partial order reduction). This approach allows to separate the aspects related to the encoding of the verification problem (performed by the application tool) and its resolution (done by the CÆSAR_SOLVE library), leading to verification tools with a highly modular architecture.

We also worked to combine functional verification and performance evaluation. We developed several performance evaluation tools that are now part of CADP 2006: BCG_MIN, DETERMINATOR, BCG_STEADY, and BCG_TRANSIENT). So far, they have been used sucessfully in two case-studies: the lifetime analysis of the Hubble space telescope and the fairness analysis of the SCSI-2 bus arbitration protocol.

### 1.7.3 Scientific achievements

**OPEN/CÆSAR Libraries.** The generic libraries provided by the OPEN/CÆSAR environment [Gar98] are useful modules for on the fly verification (e.g., state tables, stacks, transition lists, bitmap tables, etc.), playing a central role in the CADP toolbox. During 2004, two new libraries were added: CAESAR_AREA (800 lines of C code), which allows different objects (states, labels, character strings, user-defined memory blocks) to be handled uniformly, and CAESAR_MASK (1,260 lines of C code), which exports primitives for applying sequences of hiding and renaming operations (defined using regular expressions) to memory blocks, allowing in particular to hide and/or rename labels on the fly. Also, the existing CAESAR_HASH library was improved by adding new hash functions and rewriting

[Gar98] Hubert Garavel. OPEN/CÆSAR: An Open Software Architecture for Verification, Simulation, and Testing. In Bernhard Steffen, editor, *Proceedings of the First International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'98 (Lisbon, Portugal)*, volume 1384 of *Lecture Notes in Computer Science*, pages 68–84, Berlin, March 1998. Springer Verlag. Full version available as INRIA Research Report RR-3352.

several existing ones, and the Caesar_Table library was enhanced by extending the table maximal capacity from $(2^{24}) - 1$ to $2^{29}$ elements.

Another new library, named Cæsar_Solve [28, 29, 7], was developed for solving boolean equation systems of alternation depth 1 (i.e., without mutual recursion between minimal and maximal fixed point equations) on the fly. This library is at the core of several Cadp verification tools, namely the equivalence checker Bisimulator, the model checkers Evaluator 3.5 and 4.0, and the minimization tool Reductor 5.0. The resolution method is based on boolean graphs, which provide an intuitive representation of dependencies between boolean variables, and which are handled implicitly, in a way similar to the Open/Cæsar interface.

Cæsar_Solve (12, 200 lines of C code), which became part of Cadp in 2004, currently provides five different resolution algorithms. A1 and A2 are general algorithms based upon depth-first, respectively breadth-first, traversals of boolean graphs. A3 and A4, based upon memory-efficient depth-first traversals of boolean graphs, are optimized for the case of acyclic, respectively disjunctive/conjunctive, boolean graphs. A5 is a general algorithm based upon a depth-first traversal of boolean graphs; it generalizes Tarjan's algorithm for computing strongly connected components and is much faster than A1 and A2 when it is invoked many times on the same equation block. All these algorithms can generate diagnostics explaining why a result is true or false (examples and counterexamples).

**The REDUCTOR 5.0 Tool.** A mean to fight state explosion is to reduce transition systems on the fly, still preserving certain equivalence relations. The Reductor 5.0 tool (2, 000 lines of C code), which became part of Cadp in 2005, performs exhaustive reachability analysis combined with several kinds of reductions: elimination of $\tau$-transitions and the so-called *redundant* transitions [Mou92], still preserving safety equivalence; elimination of all $\tau$-transitions, still preserving $\tau^*.a$ equivalence; elimination of all circuits of $\tau$-transitions, still preserving branching equivalence (this reduction is called $\tau$-compression); reduction by $\tau$-confluence [GP00] (a form of partial order reduction preserving branching equivalence); elimination of duplicate transitions, still preserving strong equivalence; full minimization of a graph modulo strong equivalence; trace reduction (normal determinization); weak trace reduction (determinization with $\tau$-elimination); and $\tau$-divergence reduction (similar to $\tau$-compression, but keeping divergences as $\tau$-loops). The 1st, 4th, and 6th reductions are implemented using the boolean resolution algorithms provided by the Cæsar_Solve library. The 6th reduction is "orthogonal" in the sense that it can be combined with any of the other reductions. Experiments on various communication protocols and distributed systems have shown that these reductions, and $\tau$-confluence in particular, may reduce the number of states and transitions by up to 3 orders of magnitude [33, 31].

**Performance Evaluation Tools** 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, still represented

[Mou92] Laurent Mounier. *Méthodes de vérification de spécifications comportementales : étude et mise en œuvre.* Thèse de Doctorat, Université Joseph Fourier (Grenoble), January 1992.

[GP00] J.F. Groote and J. van de Pol. State Space Reduction using Partial $\tau$-Confluence. In Mogens Nielsen and Branislav Rovan, editors, *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science MFCS'2000 (Bratislava, Slovakia)*, volume 1893 of *Lecture Notes in Computer Science*, pages 383–393, Berlin, August 2000. Springer Verlag. Also available as CWI Technical Report SEN-R0008, Amsterdam, March 2000.

in the Bcg format. The Determinator, Bcg_Steady, and Bcg_Transient tools [20] (4,350 lines of C code in total) became part of Cadp in 2004, after their source code were entirely scrutinized and revised, their command-line interface were enhanced, their various output formats were improved, and their manual pages were finalized. Also, a subtle semantic bug was corrected in the algorithm used by Bcg_Min to minimize a probabilistic/stochastic model with respect to stochastic branching bisimulation.

### 1.7.4 Collaborations

- Holger Hermanns (Saarland University, Germany) contributed to the Bcg_Min, Determinator, Bcg_Steady, and Bcg_Transient tools.

### 1.7.5 External support

The contracts ArchWare, FormalFame, FormalFame Plus, Multival, and Senva provided support to tackle this objective.

### 1.7.6 Self assessment

The Open/Cæsar and Bcg software environments have been successful in providing Cadp with strong software foundations for enumerative and on the fly verification. During the evaluation period, we extended these libraries adequately, still preserving backward compatibility.

We also gave a new impulse by developing the Cæsar_Solve library, which is part of Open/Cæsar. As far as we know, Cæsar_Solve is the only generic software library for local resolution of boolean equation systems currently available to the verification community. The interface of the library was the result of a 2-year effort, and several verification tools of Cadp (Bisimulator, Evaluator, Reductor) are connected to it. We believe that the development and enhancement of Cæsar_Solve should be continued, and we plan to extend the library with new resolution algorithms targeted to efficient bisimulation checking.

## 2 Knowledge dissemination

### 2.1 Publications

| | 2003 | 2004 | 2005 | 2006 |
|---|---|---|---|---|
| PhD Thesis | | | 1 | |
| H.D.R (*) | | | | 2(***) |
| Journal | 2 | | 1 | 3 |
| Conference proceedings (**) | 8 | 8 | 7 | 5 |
| Book chapter | | | | 1 |
| Book (written) | | | | |
| Book (edited) | 1 | | | |
| Patent | | | | |
| Technical report | 2 | | 2 | 1 |
| Deliverable | 5 | 2 | | |

(*) HDR Habilitation à diriger des Recherches
(**) Conference with a program committee
(***) to be defended in 2007/2008

Indicate the major journals in the field and, for each, indicate the number of papers coauthored by members of the project-team that have been accepted during the evaluation period.

1. Theoretical Computer Science (Tcs): 1 article
2. Science of Computer Programming (Scp): 1 article
3. International Journal on Software Tools for Technology Transfer (Sttt): 1 article
4. Technique et Science Informatiques (Tsi): 1 article.

Indicate the major conferences in the field and, for each, indicate the number of papers coauthored by members of the project-team that have been accepted during the evaluation period.

1. International Conference on Tools and Algorithms for the Construction and Analysis of Systems (Tacas): 4 articles
2. International Conference on Computer Aided Verification (Cav): 1 article
3. International Conference on Algebraic Methodology and Software Technology (Amast): 2 articles
4. International Conference on Integrated Formal Methods (Ifm): 2 articles
5. International Conference on Formal Techniques for Networked and Distributed Systems (Forte): 1 article
6. International Conference on Formal Methods for Open Object-Based Distributed Systems (Fmoods): 3 articles
7. International Workshop on Formal Methods for Industrial Critical Systems (Fmics): 2 articles
8. International SPIN Workshop on Model Checking of Software (Spin): 3 articles.

## 2.2 Software — Valorization and Technology Transfer

The Vasy project-team distributes two main software tools: the Cadp toolbox (see § 1.2.3) and the Traian compiler (see § 1.2.3).

- The latest stable version of the Cadp software (see "`http://www.inrialpes.fr/vasy/cadp`") was issued in December 2006 and many successive beta-versions have been issued since the previous stable one (issued in July 2001).

  Cadp is distributed free of charge to universities and public research centers. Industrialists can obtain an evaluation license for non-commercial use during a limited period of time. In 2007, we will start distributing Cadp under commercial licenses to industrial partners.

  Up to December 22, 2006, Cadp has been licensed to 366 sites in the world (companies, research centers, universities, etc.), who signed the Cadp license agreement. During the evaluation period, we have granted licenses of Cadp for 3096 machines around the world (838 machines in 2003, 679 machines in 2004, 663 machines in 2005, 822 machines in 2006, and 94 machines in January and February 2007).

  Cadp was used by various teams in the world for analyzing industrial case studies and for developing research software based on the generic components provided by Cadp (e.g., the Bcg and Open/Cæsar environments). To cite only the works having led to scientific publications, we have currently recorded 94 case studies (see "`http://www.inrialpes.fr/vasy/cadp/case-studies`"), of which 27 were undertaken between 2003 and 2006.

  As regards other tools that could compete with Cadp 2006, we can draw the following comparative survey, based on the comprehensive list of verification tools established at the University of Brno `http://anna.fi.muni.cz/yahoda/`. In this list of 44 tools, we consider only those that are still maintained, i.e., which have released at least one new version in 2005, 2006, or early 2007. This gives 27 tools listed in the table below. For each of them, we ask four essential questions:

  - (C1) Does the modeling language used by the tool support concurrency, i.e., does it have some builtin notion of parallel composition?
  - (C2) Does the modeling language support user-defined data types such as records, unions, lists, etc. (and not only, boolean, integers, and enumerated types)?
  - (C3) Does the tool support not only model checking (all the 44 tools in the list do) but also equivalence checking, which is a standard practice in hardware verification? (in addition to being useful to compositional verification)
  - (C4) Does the tool support distributed verification, i.e. can it use the power of a cluster of machines, rather than one single machine?

| tool name | (C1) | (C2) | (C3) | (C4) |
|---|---|---|---|---|
| APMC | | | no | |
| Bandera | | | no | |
| Blast | no | | no | |
| Cadence SMV | | | no | |
| CADP 2006 | yes | yes | yes | yes |
| DiVinE | | | no | |
| DREAM | | | no | |
| Expander2 | no | | yes | no |
| HSolver | | | no | |
| Java Path Finder | | | no | |
| LTSA | | | no | |
| $\mu$Crl | yes | yes | yes | yes |
| Moped | | | no | |
| MRMC | | | no | |
| nuSMV | | | no | |
| PRISM | | | no | |
| PROD | | | no | |
| ProVer | | | no | |
| PVS | | | yes | no |
| Reactis Tester | | | no | |
| SPIN | | | no | |
| Temporal Rover | | | no | |
| TIMES | | | no | |
| TRON | | no | yes | no |
| TwoTowers | | | yes | no |
| UPPAAL | | | no | |
| VeriSoft | | | no | |

From this table, there are only two tools with four "yes" answers: Cadp 2006 and the $\mu$Crl toolset developed by our Cwi/Sen2 colleagues in the Senva associated team (Senva = Sen2 + Vasy).

In favour of Cadp, we can underline its anteriority over $\mu$Crl in a number of domains (generation of labelled transition systems, on the fly verification, parameterized boolean equation systems, graphical interfaces, distributed verification, etc.), its large user community, and the high level of integration achieved between many different verification techniques that can be combined altogether.

- The Traian compiler is distributed via the Internet: it has a dedicated Web page ("`http://www.inrialpes.fr/vasy/traian`") from which the compiler can be downloaded freely. The latest version (2.5) was issued in October 2005. Since 2003, Traian has been downloaded from 284 sites.

  We are aware only of prototype compilers for the E-Lotos language. However, other languages (such as Fdr2) already contain some features of E-Lotos (data types, for instance).

- In collaboration with the Sen2 team of Cwi (Amsterdam, The Netherlands), Vasy developed and made available to the scientific community the Vlts benchmark suite (see "`http://www.inrialpes.fr/vasy/cadp/resources/benchmark_bcg.html`").

First benchmark base of this kind, VLTS (*Very Large Transition Systems*) is a collection of 40 labeled transition systems of increasing sizes (ranging from 300 states to 34 million states). It provides a scientific criterion for a performance assessment of algorithms and tools operating on large graphs, including graph visualization software, explicit state verification software (model checkers, equivalence checkers, and minimization tools), as well as computer formats for the representation of transition systems. The VLTS benchmark suite has already been used in several scientific publications.

In addition, 29 research tools (see "`http://www.inrialpes.fr/vasy/cadp/software`"), 12 of which were developed between 2203 and 2006, are connected to CADP or developed using the programming interfaces provided by CADP. In particular, CADP is connected to the GNA (*Genetic Network Analyser*) tool (developed by the HELIX project team) and the VERCORS (*VERification of models for distributed communicating COmponents, with safety and Security*) platform (developed by the OASIS project team).

In the context of the FORMALFAME and FORMALFAME PLUS contracts, formal verification using LOTOS and CADP is now used on a daily basis by BULL. In particular, it has been used for the verification of critical parts of the cache coherency protocols in BULL's high end servers that were used to build Tera10, Europe's most powerful super computer. We continue this technology transfer in the framework of the MULTIVAL project with, in addition to BULL, ST MICROELECTRONICS and CEA/LETI.

## 2.3 Teaching

- In 2003, A. Collomb gave three courses on web sites, internet services, and algorithms at IUT 2 of Université Pierre Mendès-France (122 hours).

- In 2003, C. Joubert gave a course on functional programming to 2nd year students of Université Joseph Fourier, Grenoble (33 hours).

- In 2004, C. Joubert gave three courses on imperative programming, software and hardware architecture, and linear algebra to 2nd year computer science students of license and IUP MIAGE at Université Joseph Fourier, Grenoble (75.5 hours).

- In 2005, C. Joubert gave a course on software engineering to 4th year students and also gave lectures and programming assignments for the courses on formal specification, computer networks, software architecture, and operating systems at Université Joseph Fourier (96 hours).

- In 2003, F. Lang gave, jointly with Flavio Oquendo, a course on protocol specification and verification to the MSc computer science students of Université de Savoie (24 hours).

- In 2005 and 2006, F. Lang and W. Serwe gave a course on real-time to the 3rd year students of ENSIMAG (18 hours per year).

- In 2003 and 2004, R. Mateescu gave, jointly with A. Collomb and W. Serwe, a course on real-time to the 3rd year students of ENSIMAG (21 hours per year).

- In 2007, H. Garavel, F. Lang, and W. Serwe gave, jointly with P. Raymond, a course on formal methods for software development to the computer science engineer students of CNAM (27 hours).

- R. Mateescu supervised the internships of one student from Ecole Polytechnique and one student from EPITA (Paris).
- F. Lang and A. Collomb supervised the internship of one student from Supélec Metz.
- H. Garavel and F. Lang supervised the internships (*mémoire de probatoire* CNAM and *mémoire* CNAM) of 4 students.
- In the computer science master entitled "*Informatique: Systèmes et Logiciels*", common to Institut National Polytechnique de Grenoble and Université Joseph-Fourier (Grenoble), F. Lang and R. Mateescu supervised one student and participated to the jury of 2 other students.
- R. Mateescu participated to the MSc jury of one student at Ecole Normale Supérieure de Lyon.
- H. Garavel participated to the jury of 4 PhDs: Frédéric Tronel (Université de Rennes) in 2003, Jun Pang (Free University of Amsterdam, The Netherlands) and Nestor Cataño Collado (Université Paris 7) in 2004, and Marie Lalire (Institut National Polytechnique de Grenoble) in 2006.
- R. Mateescu participated to the jury of 5 PhDs: Rabea Boulifa (Université de Nice Sophia-Antipolis) in 2004, Jesús Martínez Cruz (University of Málaga, Spain) in 2005, Gavril Godza (Polytechnic University of Bucharest, Romania), Juan José Sánchez Penas (University of Corunha, Spain) and Emilie Oudot (Université de Franche Comté, Besançon) in 2006.
- F. Lang participated to the jury of 2 PhDs: Arnaud Lanoix (Université de Franche Comté, Besançon) and Tomas Barros (Université de Nice Sophia-Antipolis) in 2005.
- H. Garavel and R. Mateescu supervised the PhD thesis of C. Joubert (Institut National Polytechnique de Grenoble), defended in 2005.
- H. Garavel was a jury member of Leila Kloul's habilitation thesis (University of Versailles-Saint-Quentin-en-Yvelines) in 2006.
- R. Mateescu was a member of the "commission de spécialistes" (section 27) at Université de Savoie in 2003 and 2004, and at Université de Bourgogne since 2006.

The VASY project-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.

## 2.4 Visibility

### Publishing activities

- H. Garavel was, together with John Hatcliff (Kansas State University), responsible for a special issue [GH06b] of the STTT (*Software Tools for Technology Transfer*) journal, which gathers the best software-oriented papers of TACAS'2003.

[GH06b] Hubert Garavel and John Hatcliff. Why you should definitely read this special section. *Springer International Journal on Software Tools for Technology Transfer (STTT)*, 8(1):1–3, February 2006.

- H. Garavel was, together with John Hatcliff (Kansas State University), responsible for a special issue [GH06a] of the TCS (*Theoretical Computer Science*) journal, which gathers the best theory-oriented papers of TACAS'2003.

- H. Garavel contributed to paper selection for a related special issue of the FMSD (*Formal Methods in System Design*) journal, which gathered the best papers of PDMC'2004.

**Organization of workshops/conferences**

- H. Garavel was, together with John Hatcliff (Kansas State University), program chair of TACAS'2003 (*9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Varsow, Poland, April 5–13, 2003). VASY hosted the TACAS'2003 conference WEB site.

- F. Lang, A. Collomb, and W. Serwe co-organized (with Marieke Huisman, INRIA Sophia-Antipolis) the "MODOCOP *Workshop on Smart Card Specification, Verification, and Testing*", INRIA Rhône-Alpes, December 4, 2003.

- R. Mateescu organized an international seminar of the ERCIM working group FMICS devoted to the preparation of a "Formal Methods Handbook", held in Aix-les-Bains on April 20–22, 2004.

- The 1st 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`".

- The 2nd annual SENVA seminar was held in St. Pierre de Chartreuse on May 30–June 1st, 2005. The list of talks is available from "`http://www.inrialpes.fr/vasy/senva/workshop2005`".

- An international meeting on "*Clusters and Grids for Verification and Performance Evaluation*" held at INRIA Rhône-Alpes on November 16–17, 2005. The list of talks is available from "`http://www.inrialpes.fr/vasy/senva/meeting2005`".

- The 3rd annual SENVA seminar was held in Venosc on June 12–14, 2006. The list of talks is available from "`http://www.inrialpes.fr/vasy/senva/workshop2006`".

**Responsibilities in the scientific community**

- The VASY project-team is member of the FMICS (*Formal Methods for Industrial Critical Systems*) working group of ERCIM (see "`http://www.inrialpes.fr/vasy/fmics`"). Since July 2002, H. Garavel is member of the FMICS Board, in charge of dissemination actions.

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

- H. Garavel is a member of the technical committee (*ETItorial Board*) of the ETI (*Electronic Tool Integration*) software development platform (see "`http://www.eti-service.org`").

[GH06a] Hubert Garavel and John Hatcliff. TACAS 2003 Special Issue — Preface. *Theoretical Computer Science*, 354(2):169–172, March 2006.

**Program and steering committees**

- H. Garavel was a steering committee member of the Pdmc (*Parallel and Distributed Methods in Verification*) series of international workshops in 2005 and 2006.
- H. Garavel was a program committee member of 7 international conferences and workshops.
- R. Mateescu was a program committee member of 15 international conferences and workshops, and of 2 national conferences.
- F. Lang was a program committee member of 1 international conference and 1 national conference.

## 2.5 Miscellaneous

In June 2006, *Scientific American* (the most read scientific journal) mentioned Cadp in a short list of 12 tools for verification of software designs.

## 3 External Funding

| (k euros) | 2003 | 2004 | 2005 | 2006 |
|---|---|---|---|---|
| INRIA Research Initiatives | | | | |
| ARC† Modocop | 5 | — | — | — |
| National initiatives | | | | |
| RNTL Parfums | 58 | — | — | — |
| ACI Fiacre | — | 8 | 32 | 32 |
| RNTL OpenEmbeDD | — | — | — | 21 |
| European projects | | | | |
| IST ArchWare | 55 | 55 | 27 | — |
| Ec-Moan | — | — | — | — |
| Associated teams | | | | |
| Senva | — | 20 | 15 | 10 |
| Industrial contracts | | | | |
| FormalFame | 64 | 16 | — | — |
| FormalFame Plus | — | 15 | 39 | 40 |
| Topcased | — | — | — | 76 |
| Multival | — | — | — | — |
| Total | 182 | 114 | 113 | 179 |
| Scholarships | | | | |
| PhD * | 1 | 1 | 1 | 1 |
| Post Doc* | 0 | 1 | 1 | 1 |
| AI+ | 0 | 0 | 0 | 0 |
| ODL# | 0 | 0 | 0 | 0 |
| Total | 1 | 2 | 2 | 2 |

† INRIA Cooperative Research Initiatives
‡ Large-scale Initiative Actions
∗ other than those supported by one of the above projects
+ junior engineer supported by INRIA
# engineer supported by INRIA

### ARCs

- Modocop[1] (January 2002 – December 2003) was an Arc project between the Inria project-teams Landes (Rennes), Lemme (Sophia-Antipolis), Oasis (Sophia-Antipolis), Vertecs (Rennes), and Vasy. Modocop studied a framework for automatic specification, verification, and symbolic testing of concurrent object-oriented programs, combining proof techniques and model checking techniques.

### National initiatives

- Parfums[2] (May 2001 – April 2003) was a pre-competitive Rntl project, which gathered three companies (Mge-Ups, ScalAgent — a startup originating from the former Sirac project-team of Inria, and Silicomp) and Vasy. Parfums aimed at implementing a safe, flexible Java architecture enabling the remote management of uninterruptible power supplies from various equipments.

[1]http://www-sop.inria.fr/lemme/modocop
[2]http://www.rntl.org/projet/resume2000/Parfums.htm

- Fiacre[3] (started in September 2004) is a 3-year project of Aci *Sécurité Informatique* gathering the Inria project-teams Oasis (Sophia-Antipolis) and Vasy, the Ltci team of Enst-Paris, and the Svf team of the Feria-Laas laboratory (Toulouse). Fiacre aims at designing methods and tools for specification, model extraction, and verification of distributed, hierarchical, and communicating components.

- The Inria/Leti pilot research center[4] (started in March 2004) is a research initiative involving the Vasy project-team of Inria and the Leti laboratory of Cea-Grenoble. We collaborate with Leti on software tools for verifying asynchronous circuits, Gals (*Globally Asynchronous Locally Synchronous*) architectures, NoC (*Network on Chip*), and SoC (*System on Chip*).

- OpenEmbeDD[5] (started in May 2006) is a 3-years Rntl project involving four companies (Airbus, Cs, France Telecom, and Thales), the List laboratory of Cea, several project-teams of Inria (among which Vasy), the Laas laboratory (Toulouse), and the Verimag laboratory (Grenoble). OpenEmbeDD aims at developing an open-source, generic, standard software engineering platform for real-time embedded systems.

**European projects**

- ArchWare[6] (January 2002 – July 2005) was a project of the European "Information Society Technologies" program (Ist-2001-32360), gathering the Research Consortium of Pisa (Cpr), the Engineering company (Italy), the University of Savoie (Llp/Cesalp laboratory and "Association Interaction Université-Economie" — InterUnec), the Thésame company (France), the Universities of Manchester and St Andrews (United Kingdom), and the Vasy project-team of Inria. ArchWare aimed at building an integrated environment for architecting evolvable software systems with functional and performance requirements.

- Ec-Moan (started in February 2007) is a 3-year project of the European program "Tackling Complexity in Science" (Fp6-Nest-Path-Com-043235), gathering the Cwi and the Free University of Amsterdam (The Netherlands), the Masaryk University of Brno (Czech Republic), the University of Edinburgh (United Kingdom), Université Joseph Fourier (Grenoble) and the Helix and Vasy project-teams of Inria. Ec-Moan aims at developing new, scalable methods for modeling and analyzing integrated genetic, metabolic, and signaling networks, and the application of these methods for a better understanding of the Escherichia Coli bacterial model system.

**Associated teams and other international projects**

- Senva[7] is a joint research team on safety-critical systems between the Vasy project-team of Inria and the Sen2 team of Cwi (Amsterdam, The Netherlands). Launched in 2004, the Senva team is supported by Inria's European and International Affairs Department and by Cwi. The first three years of Senva have been favorably evaluated by a panel of international experts in November 2006.

[3] `http://www-sop.inria.fr/oasis/fiacre`
[4] `http://www.inrialpes.fr/vasy/doc/communique-LETI-INRIA.pdf`
[5] `http://openembedd.inria.fr`
[6] `http://www.arch-ware.org`
[7] `http://www.inrialpes.fr/vasy/pub/senva`

**Industrial contracts**

- FormalFame[8] (2001 – 2004) was a contract between Bull and Vasy. FormalFame aimed at applying formal methods and tools developed at Inria to detect design errors at the very beginning of the development process, so as to improve the quality and productivity of complex system designs. The target of FormalFame was Fame, a multiprocessor architecture developed by Bull based on Intel's 64-bit Itanium-2 processors and used in Bull's NovaScale high-end servers and in the Tera10, Europe's most powerful supercomputer.

- FormalFame Plus[9] (2004 – 2007) is a contract between Bull and Vasy. FormalFame Plus aims at enhancing the performance and usability of the Cadp tools to address the Fame-2 multiprocessor architecture under design at Bull for their future high-end servers. An important part of the FormalFame Plus contract is the development of the Lnt2Lotos translator, to ease the development of large formal specifications by Bull (see Section 1.4.3).

- TopCased[10] (started in August 2006) is a 4-years project of Aese, the French *pôle de compétitivité* dedicated to aeronautics, space, and embedded systems. This project gathers 23 partners, including companies developing safety-critical systems such as Airbus (leader), Astrium, Atos Origin, Cs, Siemens VDO, Thales Aerospace, etc. TopCased develops a modular, open-source, generic Case environment providing methods and tools for embedded system development, ranging from system and architecture specifications to software and hardware implementation through equipment definition.

- Multival[11] (started in December 2006) is a 3-year project of Minalogic, the French *pôle de compétitivité* dedicated to micro-nano technologies and embedded software for systems on chip (Emsoc cluster). Multival addresses verification and performance evaluation issues for three innovative asynchronous architectures developed by Bull, Cea/Leti, and ST Microelectronics.

[8]http://www.inrialpes.fr/vasy/dyade/formalfame.html
[9]http://www.inrialpes.fr/vasy/dyade/formalfame.html
[10]http://www.topcased.org
[11]http://www.inrialpes.fr/vasy/multival

## 4 Objectives for the next four years

In fact, the work agenda for Vasy strongly relies on four funding contracts that have been signed already, and are now starting or have just started, namely OpenEmbedd (2006-2009), Topcased (2006–2010), Multival (2006-2009), and Ec-Moan (2007-2010).

Compared to the past period (2003-2006), these new contracts will increase Vasy's external funding by a factor of 3 (approximately), thus allowing numerous software engineers to join Vasy.

The counterpart is that, to a large extent, these contracts predetermine Vasy's future research activities. Fortunately, the commitments of Vasy as regards OpenEmbedd, Topcased, Multival, and Ec-Moan fit nicely with the line of work that Vasy proposes to do.

The objectives of the Vasy project-team for 2007–2011 are also motivated by two considerations:

- There is a heavy trend towards asynchrony in computing. Due to electrical limits on silicon, one is reaching the point where the Moore law will no longer be valid. Hardware makers, who have been so far responsible for most performance improvements, can no longer increase processor clock frequencies as they used to.

  The only way to get more performance will be to go for multiprocessor systems (two-core, four-core, etc.), which means that existing software must be deeply revisited to take advantage of concurrent processing. Thus, software makers too will now be responsible for performance enhancements.

  Given the difficulties in designing correct multiprocessor architectures and correct parallel software, there will be an even stronger need for tools (such as those developed by Vasy) dealing with asynchrony.

  With respect to hardware design issues, Vasy is ideally positioned for the next years, because of its co-operation with three major French (and international) hardware makers: Bull (who develops the Fame2 multiprocessor architecture for high-end servers), Cea/Leti (who develops the innovative Faust2 network-on-chip architecture) and ST Microelectronics (who develops the xSTream multiprocessor system on chip). Vasy also collaborates with Cea/Leti to verify asynchronous circuits.

  Since we will focus on hardware verification problems, we will probably put less emphasis on investigating software issues related to parallelizing existing sequential code to benefit from multicore processors and multiprocessor architectures (although we will face similar problems when trying to parallelize parts of Cadp software itself — see below).

- At the same time, we believe that research in formal verification has reached a point where it will shift from simple prototypes to complex software tools. This is a normal evolution that corresponds to the (slow, yet real) acceptance of formal methods and tools by the industry.

  Eventually, there might be little place for formal languages not supported by tools, and for tools that are not efficient, or not usable, or provide only a functionality and that are not properly integrated into larger tool sets.

  In this respect, the research agenda of Vasy has been clear from the beginning, since we have always invested a significant effort in developing robust software tools to implement our verification algorithms. The dissemination figures for Cadp effort (see above) indicate that this effort is recognized and appreciated by a large academic community. The on-going contracts also demonstrate some industrial interest in using the technologies developed at Vasy.

For the 2007–2011 period, we wish to focus on the following research themes:

**Advanced specification languages.** Models of concurrent systems have been studied for long in concurrency theory, but with limited industrial impact. More recently, Uml and the "model-driven" approaches have succeeded in getting attention from the industry. However, this kind of models is almost syntactic and lacks the sound semantic foundations provided by concurrency theory.

We are therefore convinced that there is a clear potential for specification langagues that would be formal, expressive, and executable at the same time. We believe that process algebras are a strong basis for this, provided suitable adaptations for industrial acceptance. We plan to pursue our work in the following directions:

- Because Lotos is now used by several companies (especially in the Multival project), we must maintain some activity in this domain. Our technology for compiling Lotos is now well-established. However, it could be significantly improved by enhancing the semantical model used internally and by introducing new static analysis techniques to fight state explosion. These ideas have been tried in small prototypes aside Cadp; they need to be studied on a larger scale and eventually merged into the mainstream tools of Cadp.
- We also consider other languages than Lotos. In particular, we plan to finish our translators from various languages to Lotos.

  The Fsp2Lotos and Fdr2Lotos translators will broaden the use of Cadp in direction of the Fsp and Csp user communities, which are quite important, especially in the United Kingdom.

  We will maintain the Chp2Lotos translator, which is successfully used by Cea/Leti for designing verified circuits (for instance, in the European project Neva).

  We will continue our work on Lnt2Lotos, which is a translator from (a subset of) Lotos NT to Lotos funded by Bull to ease the development of formal specifications by non-experts. In particular, this translator provides expert knowledge for implementing classes of data structures and processes "optimally" (i.e., from a model checking point of view). This work is also of interest to our industrial partners seeking to increase penetration of formal methods in their companies.
- In parallel to Lnt2Lotos, which implements Lotos NT indirectly (by translation to Lotos), we will pursue the work undertaken on Traian, a native implementation of Lotos NT. The current version of Traian must be entirely rewritten, since it was developed using the compiler generator tool Fnc2 that is no longer maintained, and since Traian itself is used as a compiler generator for most of the Cadp compilers. The new Traian should also implement the process part of Lotos NT (not handled currently) by establishing a connection with the Ntif/Fiacre intermediate forms. Finally, timed aspects should also be dealt with, e.g., by developing connections with existing tools such as Uppaal, Tina, etc.

**Advanced verification and performance evaluation.** Another part of our work is devoted to verification techniques that are largely independent from any specification language (namely, from Lotos, Lotos NT, etc.). We plan to consider three main areas (which are not totally disjoint):

- **Distributed verification:** it consists in exploiting the computing resources (memory and Cpu time) of many machines (clusters and grids) to fight against the state explosion problem.

  Cadp was among the first (if not the first) verification toolsets to include distributed verification (namely, the Distributor and Bcg_Merge tools) and to make them available publicly to external users.

  During the recent years, we have developed prototype tools based on distributed verification (a distributed solver for boolean equation systems, a distributed version of the Evaluator model checker, a distributed version of the Bisimulator model checker, etc.)

  We will focus on finalizing and assessing these tools on large clusters and real-size examples. We will also devise new distributed algorithms operating on partitioned transition systems.

- **On-the-fly model checking:** We will pursue our work on model checking $\mu$-calculus formulas extended with data. Such formulas allow to verify useful properties that cannot be expressed easily in standard $\mu$-calculus.

  A first goal will be the completion of the Evaluator 4.0 prototype model checker, which supports $\mu$-calculus with data, and its assessment on real-case-studies, especially in comparison with the current Evaluator 3.5 model checker that handles standard $\mu$-calculus only. This work item also covers a generic connection with user-defined data types contained in the specifications under verification and studies related to parameterized boolean equation systems. We will also consider higher-alternation fragments of the modal $\mu$-calculus, able to describe accurately complex fairness properties.

  A second goal will be the design of optimizations to enhance the performance of model checking algorithms based on boolean equation systems and implemented in the Cæsar_Solve library. This includes on the fly state space reductions that depend on the modal $\mu$-calculus formulas under verification, and/or on the fly state space reductors for commonly used equivalences (strong, branching, observational, etc.), as well as vacuity detection techniques indicating whether the property specified by the user is relevant with respect to a given transition system. Such reductions can also be combined with other optimizations, such as state space caching techniques (which store a limited amount of states and use various strategies for replacing old states with newly visited ones), and with out-of-core techniques.

- **Performance evaluation:** Following a strong industrial demand from Bull and ST Microelectronics, we plan to collaborate with Saarland University (Prof. Holger Hermanns) on the combination of functional verification and performance evaluation. The performance evaluation tools already present in Cadp 2006 (Bcg_Min, Determinator, Bcg_Steady, and Bcg_Transient) will be revisited in the light of industrial case-studies, and it is likely that new tools will be developed.

**Significant case-studies.** Considering the 94 case-studies tackled so far using Cadp, it is clear that there are many classes of asynchronous systems that would be worth studying. Given our limited resources, we have to make choices and we thus plan to focus on the three key areas:

- avionics embedded software (case-study to be provided by Airbus),
- multiprocessor architectures (case-studies provided by Bull, CEA/Leti, and ST Microelectronics), and
- bio-informatics (genetic regulatory networks provided by biologists).

**High-quality software tools.** This last work item is orthogonal to the previous ones. It targets at adapting CADP and TRAIAN to the forthcoming machines and environments, and at enhancing their quality and usability, so as to maintain (and hopefully increase) our user base. It consists in many sub-goals, most of which interfere with each other. We can mention some of them:

- Finalizing prototype tools developed at VASY, but not integrated yet in CADP (this is especially the case of distributed verification tools);
- Porting CADP and TRAIAN from 32-bit to 64-bit servers (namely, Intel EM64T and AMD64 running 64-bit Linux, Intel Itanium running 64-bit Linux, and Sparc running 64-bit Solaris, and possibly Windows 64-bit and MacOS 64-bits when available); this requires deep changes as certain components of CADP have been specifically optimized under the 32-bit assumption; also, CADP and TRAIAN are built using INRIA-made compiler generators (SYNTAX and FNC2) that only exist in 32-bit mode: VASY collaborates with the SYNTAX team on a 64-bit migration; TRAIAN will have to be rewritten as FNC2 is no longer maintained;
- Inserting CADP into the latest generation of IDE (*Integrated Development Environments*); this encompasses the ECLIPSE platform (for which we have already started to develop plugins), the model-driven engineering platforms (as addressed by the OPENEMBEDD and TOPCASED projects), as well as CAD (*Computer-Aided Design*) workbenches (in this respect, we have started studying how to combine CADP with SYSTEM C/TLM);
- Improving quality control for all software components of CADP; VASY already implements good software engineering practices (version control, source code checkers, etc.); for many years, we have been using individual regression testing for the most complex components of CADP; since 2004, we are performing regression testing for the 40 demo examples of CADP every night, which allowed to catch mistakes as well as different behaviours between operating systems; we plan to set up intensive semantic testing for all components so as to catch subtle semantic mistakes that would otherwise remain undetected.

Although these goals require significant manpower resources, this will be covered by the significant increase in VASY's contractual funding for the next four years.

# 5 Bibliography of the project-team

## 5.1 Books and Monographs

[1] Hubert Garavel and John Hatcliff, editors. *Proceedings of the 9th International Conference on Tools and Algorithms for Construction and Analysis of Systems TACAS'2003 (Warsaw, Poland)*, volume 2619 of *Lecture Notes in Computer Science*. Springer Verlag, April 2003.

## 5.2 Doctoral dissertations and "Habilitation" theses

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

## 5.3 Articles in referred journals and book chapters

[3] Grégory Batt, Delphine Ropers, Hidde de Jong, Johannes Geiselmann, Radu Mateescu, Michel Page, and Dominique Schneider. Validation of Qualitative Models of Genetic Regulatory Networks by Model Checking: Analysis of the Nutritional Stress Response in Escherichia Coli. *Bioinformatics*, 21(Suppl 1):i19–i28, 2005.

[4] Hubert Garavel and Wendelin Serwe. State Space Reduction for Process Algebra Specifications. *Theoretical Computer Science*, 351(2):131–145, February 2006.

[5] Frédéric Lang. Explaining the Lazy Krivine Machine Using Explicit Substitution and Addresses. *Journal of Higher-Order and Symbolic Computation, special issue on Krivine's machine*, 2006.

[6] Radu Mateescu. Logiques temporelles basées sur actions pour la vérification des systèmes asynchrones. *Technique et Science Informatiques*, 22(4):461–495, 2003. Full version available as INRIA Research Report RR-5032.

[7] Radu 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)*, 8(1):37–56, February 2006. Full version available as INRIA Research Report RR-5948, July 2006.

[8] Radu Mateescu. *Modélisation et analyse de systèmes asynchrones avec CADP*. In Nicolas Navet, editor, *Systèmes temps reél 1 — techniques de description et de vérification*, chapter 5, pages 151–180. Traité IC2. Lavoisier, 2006. Full version available as INRIA Research Report RR 5953.

[9] Radu Mateescu and Mihaela Sighireanu. Efficient On-the-Fly Model-Checking for Regular Alternation-Free Mu-Calculus. *Science of Computer Programming*, 46(3):255–281, March 2003.

## 5.4 Publications in Conferences and Workshops

[10] Grégory Batt, Damien Bergamini, Hidde de Jong, Hubert Garavel, and Radu Mateescu. Model Checking Genetic Regulatory Networks using GNA and CADP. In Susanne Graf and Laurent Mounier, editors, *Proceedings of the 11th International SPIN Workshop on Model Checking of Software SPIN'2004 (Barcelona, Spain)*, volume 2989 of *Lecture Notes in Computer Science*, pages 156–161. Springer Verlag, April 2004.

[11] Grégory Batt, Delphine Ropers, Hidde de Jong, Johannes Geiselmann, Radu Mateescu, Michel Page, and Dominique Schneider. Analysis and Verification of Qualitative Models of Genetic Regulatory Networks: A Model-Checking Approach. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proceedings of the 19th International Joint Conference on Artificial Intelligence IJCAI'05 (Edinburgh, Scotland)*, pages 370–375, July–August 2005.

[12] Damien Bergamini, Nicolas Descoubes, Christophe Joubert, and Radu Mateescu. BISIMULATOR: A Modular Tool for On-the-Fly Equivalence Checking. In Nicolas Halbwachs and Lenore Zuck, editors, *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2005 (Edinburgh, Scotland, UK)*, volume 3440 of *Lecture Notes in Computer Science*, pages 581–585. Springer Verlag, April 2005.

[13] Carlos Canal, Pascal Poizat, and Gwen Salaün. Synchronizing Behavioural Mismatch in Software Composition. In Roberto Gorrieri and Heike Wehrheim, editors, *Proceedings of the 8th IFIP International Conference on Formal Methods for Open Object-based Distributed Systems FMOODS'2006 (Bologna, Italy)*, volume 4037 of *Lecture Notes in Computer Science*, pages 63–77. Springer Verlag, June 2006.

[14] Antonella Chirichiello and Gwen Salaün. Encoding Abstract Descriptions into Executable Web Services: Towards a Formal Development. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence WI'05 (Compiègne, France)*, pages 457–463. IEEE Press, September 2005. Extended version available as Technical Report 08-2005 of Università di Roma "La Sapienza" (DIS department).

[15] Rachid Echahed, Frédéric Prost, and Wendelin Serwe. Statically Assuring Secrecy for Dynamic Concurrent Processes. In *Proceedings of the 5th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming PPDP'2003 (Uppsala, Sweden)*, 2003.

[16] Hubert Garavel. Défense et illustration des algèbres de processus. In Zoubir Mammeri, editor, *Actes de l'Ecole d'été Temps Réel ETR 2003 (Toulouse, France)*. Institut de Recherche en Informatique de Toulouse, September 2003.

[17] Hubert Garavel and Radu Mateescu. SEQ.OPEN: A Tool for Efficient Trace-Based Verification. In Susanne Graf and Laurent Mounier, editors, *Proceedings of the 11th International SPIN Workshop on Model Checking of Software SPIN'2004 (Barcelona, Spain)*, volume 2989 of *Lecture Notes in Computer Science*, pages 150–155. Springer Verlag, April 2004.

[18] Hubert Garavel, Radu Mateescu, Damien Bergamini, Adrian Curic, Nicolas Descoubes, Christophe Joubert, Irina Smarandache-Sturm, and Gilles Stragier. DISTRIBUTOR and BCG_MERGE: Tools for Distributed Explicit State Space Generation. In Holger Hermanns and Jens Palberg, editors, *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2006 (Vienna, Austria)*, volume 3920 of *Lecture Notes in Computer Science*, pages 445–449. Springer Verlag, March–April 2006.

[19] Hubert Garavel and Wendelin Serwe. State Space Reduction for Process Algebra Specifications. In Charles Rattray, Savitri Maharaj, and Carron Shankland, editors, *Proceedings of the 10th International Conference on Algebraic Methodology and Software Technology AMAST'2004 (Stirling, Scotland, UK)*, volume 3116 of *Lecture Notes in Computer Science*, pages 164–180. Springer Verlag, July 2004.

[20] Holger Hermanns and Christophe Joubert. A Set of Performance and Dependability Analysis Components for CADP. In Hubert Garavel and John Hatcliff, editors, *Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2003 (Warsaw, Poland)*, volume 2619 of *Lecture Notes in Computer Science*, pages 425–430. Springer Verlag, April 2003.

[21] Bertrand Jeannet and Wendelin Serwe. Abstracting Call-Stacks for Interprocedural Verification of Imperative Programs. In Charles Rattray, Savitri Maharaj, and Carron Shankland, editors, *Proceedings of the 10th International Conference on Algebraic Methodology and Software Technology AMAST'2004 (Stirling, Scotland, UK)*, volume 3116 of *Lecture Notes in Computer Science*, pages 258–273. Springer Verlag, July 2004.

[22] Christophe Joubert. Distributed Model Checking: From Abstract Algorithms to Concrete Implementations. In Lubos Brim and Orna Grumberg, editors, *Proceedings of the 2nd International Workshop on Parallel and Distributed Model Checking PDMC'2003 (Boulder, Colorado, USA)*, volume 89 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2003.

[23] Christophe Joubert and Radu Mateescu. Distributed On-the-Fly Equivalence Checking. In Lubos Brim and Martin Leucker, editors, *Proceedings of the 3rd International Workshop on Parallel and Distributed Methods in Verification PDMC'2004 (London, UK)*, volume 128 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2004.

[24] Christophe Joubert and Radu Mateescu. Distributed Local Resolution of Boolean Equation Systems. In Francisco Tirado and Manuel Prieto, editors, *Proceedings of the 13th Euromicro Conference on Parallel, Distributed and Network-Based Processing PDP'2005 (Lugano, Switzerland)*, pages 264–271. IEEE Computer Society, February 2005.

[25] Christophe Joubert and Radu Mateescu. Distributed On-the-Fly Model Checking and Test Case Generation. In Antti Valmari, editor, *Proceedings of the 13th International SPIN Workshop on Model Checking of Software SPIN'2006 (Vienna, Austria)*, volume 3925 of *Lecture Notes in Computer Science*, pages 126–145. Springer Verlag, March–April 2006.

[26] Frédéric Lang. EXP.OPEN 2.0: A Flexible Tool Integrating Partial Order, Compositional, and On-the-fly Verification Methods. In Jaco van de Pol, Judi Romijn, and Graeme Smith, editors, *Proceedings of the 5th International Conference on Integrated Formal Methods IFM'2005 (Eindhoven, The Netherlands)*, Lecture Notes in Computer Science. Springer Verlag, November 2005. Full version available as INRIA Research Report RR-5673.

[27] Frédéric Lang. Refined Interfaces for Compositional Verification. In Elie Najm, Jean-François Pradat-Peyre, and Véronique Viguié Donzeau-Gouge, editors, *Proceedings of the 26th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2006 (Paris, France)*, volume 4229 of *Lecture Notes in Computer Science*, pages 159–174. Springer Verlag, September 2006. Full version available as INRIA Research Report RR-5996.

[28] Radu Mateescu. A Generic On-the-Fly Solver for Alternation-Free Boolean Equation Systems. In Hubert Garavel and John Hatcliff, editors, *Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of*

*Systems TACAS'2003 (Warsaw, Poland)*, volume 2619 of *Lecture Notes in Computer Science*, pages 81–96. Springer Verlag, April 2003. Full version available as INRIA Research Report RR-4711.

[29] Radu Mateescu. On-the-Fly Verification using CADP. In Thomas Arts and Wan Fokkink, editors, *Proceedings of the 8th International Workshop on Formal Methods for Industrial Critical Systems FMICS'2003 (Trondheim, Norway)*, volume 80 of *Electronic Notes in Theoretical Computer Science*. Elsevier, June 2003.

[30] Radu Mateescu. Model Checking for Software Architectures. In Flavio Oquendo, Brian Warboys, and Ron Morrison, editors, *Proceedings of the 1st European Workshop on Software Architecture EWSA'2004 (St Andrews, Scotland, UK)*, volume 3047 of *Lecture Notes in Computer Science*, pages 219–224. Springer Verlag, May 2004.

[31] Radu Mateescu. On-the-fly State Space Reductions for Weak Equivalences. In Tiziana Margaria and Mieke Massink, editors, *Proceedings of the 10th International Workshop on Formal Methods for Industrial Critical Systems FMICS'05 (Lisbon, Portugal)*, pages 80–89. ERCIM, ACM Computer Society Press, September 2005.

[32] Flavio Oquendo, Brian Warboys, Ron Morrison, Régis Dindeleux, Ferdinando Gallo, Hubert Garavel, and Carmen Occhipinti. ArchWare: Architecting Evolvable Software. In Flavio Oquendo, Brian Warboys, and Ron Morrison, editors, *Proceedings of the 1st European Workshop on Software Architecture EWSA'2004 (St Andrews, Scotland, UK)*, volume 3047 of *Lecture Notes in Computer Science*, pages 257–271. Springer Verlag, May 2004.

[33] Gordon Pace, Frédéric Lang, and Radu Mateescu. Calculating $\tau$-Confluence Compositionally. In Jr Warren A. Hunt and Fabio Somenzi, editors, *Proceedings of the 15th International Conference on Computer Aided Verification CAV'2003 (Boulder, Colorado, USA)*, volume 2725 of *Lecture Notes in Computer Science*, pages 446–459. Springer Verlag, July 2003. Full version available as INRIA Research Report RR-4918.

[34] Pascal Poizat, Jean-Claude Royer, and Gwen Salaün. Bounded Analysis and Decomposition for Behavioural Descriptions of Components. In Roberto Gorrieri and Heike Wehrheim, editors, *Proceedings of the 8th IFIP International Conference on Formal Methods for Open Object-based Distributed Systems FMOODS'2006 (Bologna, Italy)*, volume 4037 of *Lecture Notes in Computer Science*, pages 33–47. Springer Verlag, June 2006.

[35] Gwen Salaün, Andrea Ferrara, and Antonella Chirichiello. Negotiation among Web Services using LOTOS/CADP. In Liang-Jie Zhang, editor, *Proceedings of the European Conference on Web Services ECOWS'04 (Erfurt, Germany)*, volume 3250 of *Lecture Notes in Computer Science*, pages 198–212. Springer Verlag, September 2004. Extended version available as Technical Report 13-04 of Università di Roma "La Sapienza" (DIS department).

[36] Gwen Salaün and Wendelin Serwe. Translating Hardware Process Algebras into Standard Process Algebras — Illustration with CHP and LOTOS. In Jaco van de Pol, Judi Romijn, and Graeme Smith, editors, *Proceedings of the 5th International Conference on Integrated Formal Methods IFM'2005 (Eindhoven, The Netherlands)*, Lecture Notes in Computer Science. Springer Verlag, November 2005. Full version available as INRIA Research Report RR-5666.

[37] Gwen Salaün, Wendelin Serwe, Yvain Thonnart, and Pascal Vivet. Formal Verification of CHP Specifications with CADP — Illustration on an Asynchronous Network-on-Chip. In *Proceedings of the 13th IEEE International Symposium on Asynchronous Circuits and Systems ASYNC 2007 (Berkeley, California, USA)*. IEEE Computer Society Press, 2007. to appear.

[38] Frédéric Tronel, Frédéric Lang, and Hubert Garavel. Compositional Verification Using CADP of the ScalAgent Deployment Protocol for Software Components. In Uwe Nestmann and Perdita Stevens, editors, *Proceedings of the 6th IFIP International Conference on Formal Methods for Open Object-based Distributed Systems FMOODS'2003 (Paris, France)*, volume 2884 of *Lecture Notes in Computer Science*, pages 244–260. Springer Verlag, November 2003. Full version available as INRIA Research Report RR-5012.

## 5.5 Internal Reports

[39] David Champelovier and Hubert Garavel. Reference Manual of the Lotos NT to Lotos Translator — Version 2E. Inria/Vasy, 47 pages, 2006.

[40] Bertrand Jeannet and Wendelin Serwe. Abstracting Call-Stacks for Interprocedural Verification of Imperative Programs. Research Report 4904, INRIA, July 2003.

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

[42] Pascal Poizat and Gwen Salaün. Formal Coordination of Communicating Entities described with Behavioural Interfaces. Research Report 120-2005, LaMI, Evry, September 2005.

[43] Guillaume Schaeffer. Extension et amélioration du compilateur NTIF. Mémoire d'ingénieur en 3$^{\text{ème}}$ année, Supélec, Metz, September 2003.

## 5.6 Deliverables

[44] Ilham Alloui, Hubert Garavel, Radu Mateescu, and Flavio Oquendo. The ArchWare Architecture Analysis Language. Project Deliverable D3.1, European project IST 2001-32360 "ArchWare", January 2003.

[45] Damien Bergamini, David Champelovier, Nicolas Descoubes, Hubert Garavel, Radu Mateescu, and Wendelin Serwe. ArchWare Architecture Analysis Tool by Model-Checking. Project Deliverable D3.6b, IST Project 2001-32360 "ArchWare", June 2004.

[46] Damien Bergamini, David Champelovier, Nicolas Descoubes, Hubert Garavel, Radu Mateescu, and Wendelin Serwe. Final ArchWare Architecture Analysis Tool by Model-Checking. Project Deliverable D3.6c, IST Project 2001-32360 "ArchWare", December 2004.

[47] Alban Catry, David Champelovier, Hubert Garavel, and Radu Mateescu. Definition of the Architecture Analysis Formalism for Model-Checking. Project Deliverable D3.3, European project IST 2001-32360 "ArchWare", June 2003.

[48] Alban Catry, David Champelovier, Hubert Garavel, and Radu Mateescu. Preliminary ArchWare Architecture Analysis Tool by Model-Checking. Project Deliverable D3.6a, European project IST 2001-32360 "ArchWare", December 2003.

[49] David Champelovier, Paolo Fabriani, and Hubert Garavel. Formal Specification of Federated Knowledge Management System (FKMS) using a Process Algebra. Working document, IST Project 2001-32360 "ArchWare", February 2003.

[50] Hubert Garavel and Radu Mateescu. Enhanced Model-Checker for Architecture Analysis. Project Deliverable D3.8, European project IST 2001-32360 "ArchWare", January 2003.

## 5.7 Miscellaneous

[51] Wan Fokkink, Hubert Garavel, and Jaco van de Pol. CWI and INRIA join Forces on Safety Critical Systems. *Ercim News*, (58), July 2004.