INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

An overview of CADP 2001

Hubert Garavel — Frédéric Lang — Radu Mateescu

N° 0254

Décembre 2001

THÈME 1

Rapport technique

ISSN 0249-0803

An overview of CADP 2001

Hubert Garavel* , Frédéric Lang† , Radu Mateescu‡

Thème 1 — Réseaux et systèmes
Projet VASY

Rapport technique n° 0254 — Décembre 2001 — 15 pages

Abstract: CADP is a toolbox for specifying and verifying asynchronous finite-state systems described using process algebraic languages. It offers a wide range of state-of-the-art functionalities assisting the user throughout the design process: compilation, rapid prototyping, interactive and guided simulation, verification by equivalence/preorder checking and temporal logic model-checking, and test generation. The languages, models, and verification techniques used in CADP have a broad application domain, allowing to deal with communication protocols, distributed systems, embedded software, mobile telephony, asynchronous hardware, cryptography, security, human-computer interaction, etc. CADP is currently used both in industrial companies and academic institutions for research and teaching purposes. During the last years, over 50 applications and case-studies performed using CADP have been reported.

Key-words: bisimulation – labeled transition system – LOTOS – model-checking – specification – temporal logic – verification

* `Hubert.Garavel@inria.fr`
† `Frederic.Lang@inria.fr`
‡ `Radu.Mateescu@inria.fr`

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Un aperçu de CADP 2001

Résumé : Cadp est une boîte à outils pour spécifier et vérifier des systèmes d'états finis asynchrones décrits par des langages de processus algébriques. Cadp offre de nombreuses fonctionnalités aidant l'utilisateur tout au long du procédé de conception: compilation, prototypage rapide, simulation interactive et guidée, vérification par équivalence, par pré-ordres et par évaluation de formules de logique temporelle et génération de tests. Les langages, modèles et techniques de vérification utilisées dans Cadp ont un large champ d'application, couvrant les protocoles de communication, les systèmes distribués, le logiciel embarqué, la téléphonie mobile, les circuits asynchrones, la cryptographie, la sécurité, l'interaction homme-machine, etc. Cadp est utilisé à la fois dans des sociétés industrielles et dans des institutions académiques à des fins de recherche et d'enseignement. Nous avons dénombré plus de 50 applications et études de cas réalisées ces dernières années avec Cadp.

Mots-clés : bisimulation – logique temporelle – Lotos – model-checking – spécification – système de transitions étiquetées – vérification

1 Introduction

CADP (the CÆSAR/ALDÉBARAN Development Package[1]) is a toolbox for protocol engineering, which offers a wide range of functionalities, from interactive simulation to the most recent formal verification techniques. CADP is dedicated to the efficient compilation, simulation, formal verification, and testing of descriptions written in the ISO language LOTOS [19], a value passing process algebra. It also accepts other input languages such as finite state machines and networks of communicating finite state machines.

July 2001 has seen the release of the new version CADP 2001 "Ottawa"[2]. Among many other features, CADP provides several tools for computing bisimulations (minimizations and comparisons), several model-checkers for various temporal logics and μ-calculus, and several verification algorithms including exhaustive verification, on-the-fly verification, symbolic verification using Binary Decision Diagrams, and compositional verification based on refinement. It contains many improvements and five new tools.

The architecture of CADP 2001 is displayed in Figure 1 and explained throughout the paper, which is organized as follows. Section 2 introduces the different languages and models used by the CADP tools. Section 3 describes the main tools of CADP. Section 4 presents the new tools contained in CADP 2001. Finally, Section 5 gives concluding remarks.

2 Description languages and intermediate models

The CADP toolbox accepts three different input formalisms, materialized by the three grey boxes in Figure 1:

- high-level protocol descriptions written in the ISO language LOTOS [19]: CADP contains two compilers (CÆSAR and CÆSAR.ADT) which translate LOTOS descriptions into C code that can be used for simulation, verification, and testing purposes;
- low-level protocol descriptions specified as Labeled Transition Systems (LTSs, for short), i.e., finite state machines the transitions of which are labeled by action names;
- intermediate-level networks of communicating LTSs, i.e., finite state machines running in parallel and synchronizing together by means of rendezvous; these networks can be expressed in two different formats: EXP (LTSs combined together using LOTOS parallel composition and hiding operators) and FC2 (LTSs combined together using a synchronization product).

The latest releases of the CADP toolbox devote a growing importance to the concept of intermediate formats and programming interfaces. In the sequel of this section, we present

[1] Detailed information is available at `http://www.inrialpes.fr/vasy/cadp`.

[2] CADP 2001 "Ottawa" was named in honor of Professor Luigi Logrippo and his research team at the University of Ottawa, who are actively promoting formal methods, especially LOTOS, in the telecommunication industry.

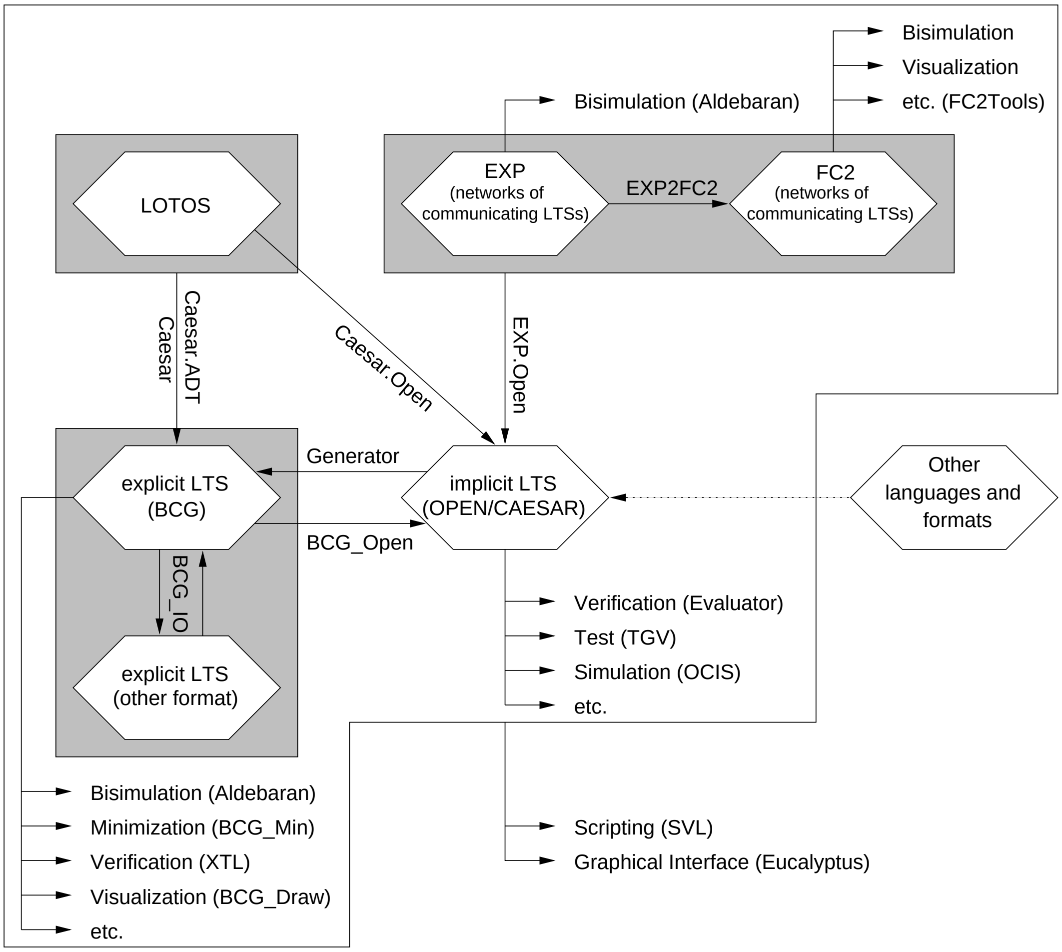

Figure 1: Architecture of CADP 2001 "Ottawa"

the OPEN/CÆSAR environment, which allows the CADP tools to be applied to protocol descriptions written in other languages than LOTOS (e.g., μCRL, SDL, UML/RT), and the BCG environment, which provides a compact LTS description format together with efficient and useful tools and libraries.

2.1 The OPEN/CAESAR environment

OPEN/CÆSAR [13] is an extensible, language-independent Application Programming Interface (API) that allows user-defined programs for simulation, execution, verification (partial, on-the-fly, etc.), and test generation to be developed in a simple and modular way. Various modules have already been written in the OPEN/CÆSAR framework, including: EVALUATOR, an on-the-fly model-checker (Section 4.2), OCIS, an interactive simulator with X-window interface (Section 4.3), TGV, a tool for the generation of conformance test suites based on verification technology (Section 4.5), and many other tools for random execution, deadlock detection, reachability analysis, sequence searching, abstraction of an LTS w.r.t. an interface, etc.

Basically, the OPEN/CÆSAR environment offers primitives to transform a system description into an LTS represented *implicitly* by its initial state and its successor function. Then, every tool connected to the OPEN/CÆSAR environment can take the resulting implicit LTS as input. Three languages have access to the OPEN/CÆSAR environment, namely the BCG graph format (Section 2.2), the LOTOS language, and networks of communicating automata in the EXP format. Since OPEN/CÆSAR is open and well-documented, users can easily extend the environment by adding their own modules or connect their own languages to fit specific needs.

2.2 The BCG graph format and libraries

BCG (*Binary-Coded Graphs*) [11] is both a format for the representation of explicit LTSs and a collection of libraries and programs dealing with this format. Compared to ASCII-based formats for LTSs, the BCG format uses a binary representation with compression techniques resulting in much smaller (up to 20 times) files. BCG is independent from any source language but keeps track of the objects (types, functions, variables) defined in the source programs.

The BCG format supports tools for drawing BCG graphs with an automatic layout of states and transitions, editing the display of BCG graphs interactively, providing information about BCG graphs such as the size of the graph, its number of states and transitions or the list of its labels, performing conversions between the BCG format and a dozen of other formats, hiding and renaming the labels of a graph according to regular expressions, generating dynamic libraries for BCG graphs, minimizing graphs according to strong or branching bisimulation (see the BCG_MIN tool, Section 4.1), etc.

Simple application programming interfaces are available to read and to produce a BCG graph. Moreover, the BCG_OPEN tool establishes a gateway between the BCG format (explicit LTSs) and the OPEN/CÆSAR environment (implicit LTSs).

3 Main tools of CADP

The Cadp toolbox contains several tools. In the sequel, we describe the most significant of these tools.

3.1 The ALDÉBARAN tool for computing bisimulations

Jointly developed by the Vasy team and the Verimag laboratory, Aldébaran [6] is a tool for verifying communicating systems, represented by Ltss. It allows the reduction of Ltss modulo various equivalence relations (such as strong bisimulation, observational equivalence, $\tau^{*}.a$ bisimulation, branching bisimulation, safety equivalence, etc.). It also allows to perform comparison according to strong bisimulation preorder, $\tau^{*}.\ a$ preorder, or safety preorder.

The verification algorithms used in Aldébaran are based either on the Paige-Tarjan algorithm for computing the relational coarsest partition [26], on the "on-the-fly" techniques proposed by Fernandez-Mounier [7], or on symbolic Lts representations using Binary Decision Diagrams (Bdds) [3]. Aldébaran has diagnosis capabilities that provide the user with explanations (counter-example sequences) when two Ltss are found to be not equivalent.

3.2 The CÆSAR compiler

Cæsar [9] is a compiler that translates the behavioral part of a LOTOS specification into either a C program (to be executed or simulated) or into an Lts.

Cæsar translation algorithms proceed in several steps. First the Lotos description is translated into a simplified process algebra called SubLotos. Then an intermediate Petri Net model is generated, which provides a compact, structured and user-readable representation of both the control and data flow. Eventually the Lts is produced by performing reachability analysis on the Petri Net.

Cæsar accepts full Lotos with very slight contextual restrictions as regards process recursion. Despite these restrictions, the subset of Lotos handled by Cæsar is large and usually sufficient for real-life needs.

The current version of Cæsar allows the generation of large Ltss (some million states) within a reasonable lapse of time. Moreover, the efficient compiling algorithms of Cæsar can also be exploited in the framework of the Open/Cæsar environment.

The most recent version of Cæsar provides a functionality called Exec/Cæsar [15] for C code generation. This C code interfaces with the real world, and can be embedded in applications. This allows rapid prototyping directly from the Lotos specification.

3.3 The CÆSAR.ADT compiler

Cæsar.adt [10] is a compiler that translates the data part of Lotos specifications into libraries of C types and functions. Each Lotos sort is translated into an equivalent C type and each Lotos operation is translated into an equivalent C function (or macro-definition).

CÆSAR.ADT also generates C functions for comparing and printing abstract data types values, as well as iterators for the sorts of finite domain.

CÆSAR.ADT accepts full LOTOS with the following (quite natural) restriction, as regards the data part: constructor operations must be identified; equations are oriented; there is a decreasing priority between equations; equations between constructors are not allowed. Also, parameterized types are not compiled (yet).

CÆSAR.ADT is fast: translation of large programs (several thousands of lines) is usually achieved in a few seconds. CÆSAR.ADT can be used in conjunction with CÆSAR, but it can also be used separately to compile and execute efficiently large abstract data types descriptions.

3.4 The XTL model-checker

XTL (*eXecutable Temporal Language*) [23] is a functional-like programming language designed to allow an easy, compact implementation of various temporal logic operators. These operators are evaluated over an LTS encoded in the BCG format. Besides the usual predefined types (booleans, integers, etc.), the XTL language defines special types, such as sets of states, transitions, and labels of the LTS. It offers primitives to access the informations contained in states and labels, to obtain the initial state, and to compute the successors and predecessors of states and transitions. The temporal operators can be easily implemented using these functions together with recursive user-defined functions working with sets of states and/or transitions of the LTS. A compiler for XTL has been developed, and several temporal logics like HML [18], CTL [4], ACTL [25], and LTAC [28] have been easily implemented in XTL.

3.5 The EUCALYPTUS graphical user interface

EUCALYPTUS [12] is a graphical user interface written in Tcl/Tk that integrates the CADP tools in a unified, user-friendly interface. This interface has the name of the project within which it was developed: the Euro-Canadian project "EUCALYPTUS".

Additionally, EUCALYPTUS integrates complementary software such as the APERO data type pre-processor for LOTOS [27], the ELUDO simulator of LOTOS descriptions [31], or the FC2 tools [1], together with the graphical editor AUTOGRAPH [29].

4 New tools of CADP 2001

The new release of CADP contains five new tools: BCG_MIN, EVALUATOR 3.0, OCIS, SVL, and TGV.

4.1 The new BCG_MIN tool for computing bisimulations

Jointly developed by the VASY team and Holger Hermanns (University of Twente), BCG_MIN implements various minimization algorithms for graphs encoded in the BCG format. It can be used to minimize "standard" LTSs, as well as "probabilistic" and "stochastic" LTSs, which may carry respectively probabilistic and stochastic labels and generalize many theoretical models published in the literature (for instance the Discrete Time Markov Chains and the Continuous Time Markov Chains).

Compared to former LTS minimization tools (including ALDÉBARAN and FC2 tools), BCG_MIN only implements two equivalences, namely strong and branching bisimulation. For these two equivalences, however, it offers compelling advantages:

- BCG_MIN can handle larger LTSs, at least larger by an order of magnitude; for instance, the largest LTS reduced so far by BCG_MIN has more than 7 million states and 40 million transitions; according to Prof. Jan Friso Groote [17], BCG_MIN is "*the best implementation of the standard (i.e., Groote & Vaandrager [16]) algorithm for branching bisimulation*";
- BCG_MIN uses BCG as its native format, thus leading to speed improvement, because BCG occupies much less disk space than most graph formats;
- BCG_MIN is able to print state equivalence classes in a user-friendly way, by relating the state numbers of the minimized graph to the state numbers of the original graph; in the case of branching equivalence, the τ-cycles are properly displayed.

4.2 The new EVALUATOR 3.0 model-checker

EVALUATOR 3.0 [24] is a new version of the EVALUATOR tool, which performs on-the-fly verification of regular alternation-free μ-calculus formulas on LTSs represented implicitly according to the OPEN/CÆSAR API. Compared to the previous version 2.0 [8], EVALUATOR 3.0 brings major improvements:

- The input specification language of EVALUATOR 3.0 is more powerful than the one of EVALUATOR 2.0:
  - action formulas can contain any combination of boolean operators and basic predicates over transition labels (which can be now given also as UNIX regular expressions over character strings);
  - regular transition sequences can be succinctly described using regular formulas built from action formulas and the usual regular expression operators;
  - it is also possible to define macro operators parameterized by formulas and to group them into separate libraries that may be included in the main specification.

- The model-checking algorithm of EVALUATOR 3.0 uses a new on-the-fly boolean resolution algorithm, which has a much better average complexity than the algorithm used in EVALUATOR 2.0. It explores less states before deciding the truth value of the formula, which leads sometimes to dramatic reductions (several orders of magnitude) of the execution time. Moreover, EVALUATOR 3.0 has been optimized in order to work more efficiently when verifying temporal formulas on explicit LTSs encoded as BCG files. Due to these optimizations, the memory consumption and the execution time of EVALUATOR 3.0 have been reduced by up to 5% and 20%, respectively.

- The diagnostics generated by EVALUATOR 3.0 are improved [22]. Diagnostics are portions of LTSs explaining either the satisfaction or the refutation of a formula: if the formula is false, a diagnostic is a counter-example; if the formula is true, a diagnostic is an example. In particular, the diagnostics obtained for derived "pure" branching-time logics like CTL and ACTL fully explain the semantics of their operators. EVALUATOR 3.0 may also serve to search regular execution sequences in the LTS, by asking for diagnostics of regular modalities.

Three libraries are also available that encode the operators of the ACTL temporal logic as well as a set of generic temporal property patterns defined by Prof. Matthew Dwyer from Kansas State University [5].

4.3 The new OCIS interactive graphical simulator

A new interactive, graphical simulator named OCIS (OPEN/CÆSAR *Interactive Simulator*) was added to CADP. Designed to replace the venerable XSIMULATOR, OCIS enables visualization and error detection during the design phase of systems containing parallelism and asynchronous communication between tasks. Its main features are:

- visualization of simulation scenarios as execution traces, trees, or Message Sequence Charts (MSCs),

- manipulation of simulation scenarios, which can be edited, saved as BCG graphs, and loaded again during another simulation session,

- manual (step by step) and automatic (pattern-guided) navigation in the system under simulation,

- source-level debugging, with access to parallel tasks, state variables, etc.

- possibility to modify the source code and to re-compile it without leaving the current simulation session.

OCIS was designed to be as much as possible language-independent and should therefore be usable for any specification language or formalism interfaced with the OPEN/CÆSAR API.

4.4 The new SVL scripting language

A new tool named Svl (*Script Verification Language*) [14] was added to Cadp. The Svl language and its associated compiler target at simplifying and automating the verification of Lotos programs. Svl behaves as a tool-independent coordination language on top of the Cadp and Fc2 tools, in the same way as Eucalyptus is a tool-independent graphical user interface.

Svl offers high-level operators for generation, parallel composition, minimization, label hiding, label renaming, abstraction, comparison, and model-checking of Ltss. It supports several methods of verification (e.g., enumerative, compositional, and on-the-fly), which can be easily combined together.

A compiler for Svl has been developed, which translates an Svl verification scenario into a Bourne shell script, which will perform all the operations needed to execute the verification scenario, e.g., invoking verification tools with appropriate options and parameters, generating intermediate files, etc.

Svl has been used in several case-studies: most of the Cadp demo examples (19 demos over a total of 29) take advantage of Svl readability and conciseness. In most cases, Svl allows the user to get rid of Makefiles and shell-scripts as well as many auxiliary files which are generated automatically from a simple Svl script. Because of its expressiveness and robustness, Svl subsumes totally the Des2Aut tool [21] used in previous versions of Cadp.

4.5 The new TGV test generator

The latest version of the Tgv (*Test Generation based on Verification technology*) [20] tool, jointly developed by the Pampa team of Inria Rennes/Irisa and the Verimag laboratory, has been integrated into the Cadp toolbox. Tgv is a tool for the automatic generation of test suites from formal specifications. These test suites are used to assess the conformance of a protocol implementation with respect to the formal specification of this protocol. Tgv takes two main inputs:

- a specification of the protocol's behavior, defined as an implicit Lts using the Open/Cæsar Api (this Api allows Tgv to be used for various languages: Lotos, Sdl, Uml/Rt, etc.),
- a test purpose, which selects the subset of the protocol's behavior to be tested; the test purpose is defined as an explicit Lts, the states of which are either normal states, accepting states (i.e., final states characterizing parts of the protocol satisfying the test purpose), or refusing states (i.e., final states characterizing parts of the behavior that are irrelevant to the test purpose).

To produce conformance test suites automatically, Tgv applies algorithms coming from verification technology. Test generation is done "on-the-fly" on the synchronous product of the specification with the test purpose; this product allows to avoid state explosion by exploring only the subset of the protocol specification permitted by the test purpose. The test cases generated by TGV are LTSs, the transitions of which carry test verdicts such as "pass", "fail", and "inconclusive".

5 Conclusion

CADP contains a lot of tools and offers a wide variety of functionalities. CADP 2001 "Ottawa" provides several of the best algorithms for simulation and verification. It supports libraries that make the addition of new tools and the connection to new languages and description formats extremely modular. It also contains a graphical user interface and a scripting language that make its use easier for both expert and non-expert users.

Moreover, CADP is supported, maintained and constantly improved. It is available on LINUX, SOLARIS, and WINDOWS platforms. It is widely distributed: in June 2001, it had been licensed to 239 sites and during year 2000, licenses were granted for 770 machines around the world; from January 1st, to June 26th, 2001, licenses were granted for 797 machines. Additionally, the CADP tools are integrated into the Web-based, open communication platform ETI (Electronic Tool Integration Platform) [30, 2].

During the last years, over 50 applications and case-studies performed using CADP have been published[3], and 10 research tools based upon the OPEN/CÆSAR and BCG environments of CADP have been developed[4]

Acknowledgements

We would like to thank all the people who contributed to the development of CADP:

- Moez Cherif, Hubert Garavel, Marc Herbert, Bruno Hondelatte, Pierre Kessler, Frédéric Lang, Stéphane Martin, Radu Mateescu, Aldo Mazilli, Frédéric Perret, Mihaela Sighireanu, and Irina Smarandache of the VASY project at INRIA Rhône-Alpes (Grenoble, France),
- Laurent Mounier and Aline Sénart of the VERIMAG laboratory (Grenoble, France),
- Thierry Jéron, Pierre Morel, and Séverine Simon of the PAMPA project at INRIA/IRISA (Rennes, France),
- Holger Hermanns at the University of Twente (The Netherlands).

We are also extremely grateful to all the scientists who contributed to the development of CADP in the past and who provided us with valuable feedback and advices about the use of CADP.

At last, we thank Solofo Ramangalahy for his useful comments about this paper.

[3]Information is available at `http://www.inrialpes.fr/vasy/cadp/case-studies`.
[4]Information is available at `http://www.inrialpes.fr/vasy/cadp/software`.

References

[1] Amar Bouali, Annie Ressouche, Valérie Roy, and Robert de Simone. The Fc2Tools set: a Toolset for the Verification of Concurrent Systems. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th Conference on Computer-Aided Verification (New Brunswick, New Jersey, USA)*, volume 1102 of *Lecture Notes in Computer Science*. Springer Verlag, August 1996.

[2] Volker Braun, Jürgen Kreileder, Tiziana Margaria, and Bernhard Steffen. The ETI Online Service in Action. In Rance Cleaveland, editor, *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS 1999)*, volume 1579 of *Lecture Notes in Computer Science*, pages 439–443, Amsterdam (The Netherlands), 1999.

[3] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8), 1986.

[4] E. Clarke, E. A. Emerson, and A. P. Sistla. Automatic Verification of Finite-State Concurrent Systems using Temporal Logic. In *10th Annual Symposium on Principles of Programming Languages*. ACM, 1983.

[5] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in Property Specifications for Finite-State Verification. In *Proceedings of the 21st International Conference on Software Engineering ICSE'99 (Los Angeles, CA, USA)*, May 1999.

[6] Jean-Claude Fernandez. An Implementation of an Efficient Algorithm for Bisimulation Equivalence. *Science of Computer Programming*, 13(2–3):219–236, May 1990.

[7] Jean-Claude Fernandez and Laurent Mounier. "On the Fly" Verification of Behavioural Equivalences and Preorders. In K. G. Larsen and A. Skou, editors, *Proceedings of the 3rd Workshop on Computer-Aided Verification (Aalborg, Denmark)*, volume 575 of *Lecture Notes in Computer Science*, Berlin, July 1991. Springer Verlag.

[8] Jean-Claude Fernandez and Laurent Mounier. A Local Checking Algorithm for Boolean Equation Systems. Rapport SPECTRE 95-07, VERIMAG, Grenoble, March 1995.

[9] Hubert Garavel. *Compilation et vérification de programmes LOTOS*. Thèse de Doctorat, Université Joseph Fourier (Grenoble), November 1989.

[10] 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.

[11] Hubert Garavel. Binary Coded Graphs: Definition of the BCG Format. Rapport SPECTRE C28, Laboratoire de Génie Informatique — Institut IMAG, Grenoble, January 1991.

[12] Hubert Garavel. An Overview of the Eucalyptus Toolbox. In Z. Brezočnik and T. Kapus, editors, *Proceedings of the COST 247 International Workshop on Applied Formal Methods in System Design (Maribor, Slovenia)*, pages 76–88. University of Maribor, Slovenia, June 1996.

[13] 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.

[14] 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.

[15] 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.

[16] Jan Friso Groote and Frits Vaandrager. An Efficient Algorithm for Branching Bisimulation and Stuttering Equivalence. In M. S. Patterson, editor, *Proceedings of the 17th ICALP (Warwick)*, volume 443 of *Lecture Notes in Computer Science*, pages 626–638. Springer Verlag, 1990.

[17] J.F. Groote and J.C. van Pol. State space reduction using partial tau-confluence. Research Report SEN-R0008, CWI, Amsterdam, The Netherlands, 2000.

[18] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32:137–161, 1985.

[19] ISO/IEC. LOTOS — A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. International Standard 8807, International Organization for Standardization — Information Processing Systems — Open Systems Interconnection, Genève, September 1988.

[20] T. Jéron and P. Morel. Test generation derived from model-checking. In N. Halbwachs and D. Peled, editors, *Proceedings of the Conference on Computer-Aided Verification CAV'99 (Trento, Italy)*, volume 1633 of *Lecture Notes in Computer Science*, pages 108–122. Springer Verlag, July 1999.

[21] 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.

[22] Radu Mateescu. Efficient Diagnostic Generation for Boolean Equation Systems. In Susanne Graf and Michael Schwartzbach, editors, *Proceedings of 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2000 (Berlin, Germany)*, volume 1785 of *Lecture Notes in Computer Science*, pages 251–265. Springer Verlag, March 2000. Full version available as INRIA Research Report RR-3861.

[23] Radu Mateescu and Hubert Garavel. XTL: A Meta-Language and Tool for Temporal Logic Model-Checking. In Tiziana Margaria, editor, *Proceedings of the International Workshop on Software Tools for Technology Transfer STTT'98 (Aalborg, Denmark)*, pages 33–42. BRICS, July 1998.

[24] Radu Mateescu and Mihaela Sighireanu. Efficient On-the-Fly Model-Checking for Regular Alternation-Free Mu-Calculus. In Stefania Gnesi, Ina Schieferdecker, and Axel Rennoch, editors, *Proceedings of the 5th International Workshop on Formal Methods for Industrial Critical Systems FMICS'2000 (Berlin, Germany)*, GMD Report 91, pages 65–86, Berlin, April 2000. Also available as INRIA Research Report RR-3899.

[25] 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.

[26] Robert Paige and Robert E. Tarjan. Three Partition Refinement Algorithms. *SIAM Journal of Computing*, 16(6):973–989, December 1987.

[27] Charles Pecheur. *Improving the Specification of Data Types in* LOTOS. Doctorate thesis, University of Liège, November 1996. Collection of Publications of the Faculty of Applied Sciences, Nr 171.

[28] Jean-Pierre Queille and Joseph Sifakis. Fairness and Related Properties in Transition Systems — A Temporal Logic to Deal with Fairness. *Acta Informatica*, 19:195–220, 1983.

[29] Valérie Roy and Robert de Simone. Auto/Autograph. In R. P. Kurshan and E. M. Clarke, editors, *Proceedings of the 2nd Workshop on Computer-Aided Verification (Rutgers, New Jersey, USA)*, volume 3 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 477–491. AMS-ACM, June 1990.

[30] Bernhard Steffen, Tiziana Margaria, and Volker Braun. The Electronic Tool Integration Platform: Concepts and Design. *Springer International Journal on Software Tools for Technology Transfer (STTT)*, 1–2(1):9–30, December 1997.

[31] B. Stepien, J. Tourrilhes, and J. Sincennes. ELUDO: The University of Ottawa LOTOS Toolkit. Technical report, University of Ottawa, 1994. Obtainable by FTP on lotos.csi.uottawa.ca.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803