HAL
open science

CADP 2006: A Toolbox for the Construction and Analysis of Distributed Processes

Hubert Garavel, Frédéric Lang, Radu Mateescu, Wendelin Serwe

To cite this version:

Hubert Garavel, Frédéric Lang, Radu Mateescu, Wendelin Serwe. CADP 2006: A Toolbox for the Construction and Analysis of Distributed Processes. Computer Aided Verification (CAV'2007), Jul 2007, Berlin, Germany. pp.158-163. inria-00189021

HAL Id: inria-00189021

https://inria.hal.science/inria-00189021v1

Submitted on 19 Nov 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

CADP 2006: A Toolbox for the Construction and Analysis of Distributed Processes

Hubert Garavel, Radu Mateescu, Frédéric Lang, and Wendelin Serwe

INRIA
Centre de recherche Rhône-Alpes / VASY
655, avenue de l'Europe, Montbonnot, 38 334 Saint Ismier Cedex, France
{Hubert.Garavel,Radu.Mateescu,Frederic.Lang,Wendelin.Serwe}@inria.fr

1 Introduction

CADP (*Construction and Analysis of Distributed Processes*)[1] [2, 3] is a toolbox for specification, rapid prototyping, verification, testing, and performance evaluation of asynchronous systems (concurrent processes with message-passing communication). The developments of CADP during the last five years led to a new release named CADP 2006 "*Edinburgh*" (as a tribute to the achievements in concurrency theory of the Laboratory for Foundations of Computer Science) that supersedes the previous version CADP 2001.

2 Modular Integration of Verification Techniques

CADP 2006 includes a complete range of functionalities for the design of asynchronous systems: code generation, rapid prototyping, random, interactive, or guided simulation, explicit-state verification, test case generation, and performance evaluation. CADP accepts as input either networks of communicating automata or higher-level specification languages, such as the ISO standard LOTOS.

Many CADP tools operate on Labeled Transition Systems (LTSs), which are represented either *explicitly*, as compact binary files encoded in the BCG (*Binary Coded Graphs*) format, or *implicitly*, as C programs implementing the transition relation according to the OPEN/CÆSAR API (*Application Programming Interface*). Three forms of verification are supported by CADP: *visual checking* (graphical inspection of an LTS), *model checking* (satisfaction of a modal μ-calculus formula by an LTS), and *equivalence checking* (comparison of two LTSs with respect to some equivalence/preorder relation).

To address the state space explosion problem, CADP 2006 provides the following verification techniques[2]:

- *Compositional verification* builds the LTS of a concurrent system incrementally by generating, minimizing, and recomposing the LTSs of individual processes. Refined compositional verification uses interface constraints (specified

[1] http://www.inrialpes.fr/vasy/cadp
[2] Related verification tools are listed at http://anna.fi.muni.cz/yahoda

manually or synthesized automatically) that restrict the behaviour of a process depending on its environment, thus limiting the size of intermediate LTSs.
- *On-the-fly verification* avoids the complete construction of an LTS by exploring only the portion relevant for verification. Model checking and equivalence checking are reformulated in terms of Boolean Equation Systems (BESs), which are solved on-the-fly using specialized linear-time algorithms.
- *Partial order reductions*, performed on-the-fly, reduce the LTS size by eliminating redundancies arising from the interleaving of independent transitions, still preserving various equivalence relations (branching bisimulation, weak trace equivalence, etc.).
- *Static analysis* aims at reducing the size of the LTSs generated from a system description, still preserving strong bisimulation.
- *Massively parallel verification* handles very large LTSs by using the computing resources of machine clusters and grids.

A key feature of CADP is to allow all these techniques to be combined in a highly modular way within the same software environment. Three examples of such combinations that cumulate the reductions and scale up to larger systems are:

1. One can use a grid to generate the LTS corresponding to a LOTOS specification, while applying static analysis and on-the-fly τ-confluence reduction simultaneously.
2. One can model check on-the-fly whether a μ-calculus formula is satisfied by a network of communicating LTSs, while applying partial order reduction.
3. One can compare on-the-fly an LTS to a network of communicating LTSs that have been previously generated (taking into account refined interface constraints) and minimized for some bisimulation.

The tools of CADP can be invoked either interactively, by using the EUCALYPTUS graphical user interface, or in batch mode, by describing the desired verification scenario as a script in the SVL language.

3 New and Enhanced Tools in CADP 2006

CADP 2006 offers 42 tools and 20 generic software libraries dedicated to verification. Numerous tools existing in CADP 2001 were entirely rewritten or significantly improved, and 15 new tools and code libraries were added:

- BCG_GRAPH generates several useful kinds of LTSs in the BCG format, such as bags, FIFO queues, and chaos LTSs.
- BCG_MERGE [5] produces a single BCG file from a PBG file (see the DISTRIBUTOR tool below). It is equipped with a graphical tool to monitor the LTS generation interactively.

- BCG_STEADY and BCG_TRANSIENT [7] perform steady-state and transient analysis of an (extended) CTMC (*Continuous-Time Markov Chain*) encoded in the BCG format.
- BISIMULATOR [11, 1] compares two LTSs on-the-fly modulo one out of 14 behavioural relations (strong, branching, observational, $\tau^*.a$, safety, trace, or weak trace equivalences — and their associated preorders). The equivalence checking problem is reformulated as a BES, which is solved using the CÆSAR_SOLVE library below. When both LTSs are not related, BISIMULATOR generates a counterexample, i.e., an acyclic LTS containing distinguishing transition sequences. Compared with former CADP tools, BISIMULATOR is more efficient and generates smaller counterexamples.
- CÆSAR 7.0 [6] is a compiler for the behaviour part of LOTOS. Among other improvements, CÆSAR 7.0 implements a static analysis based on live variable analysis, which assigns a canonical value to variables that are no longer used, thus avoiding to distinguish states that only differ by values of variables not used in the future. Compared to its previous version 6.2, CÆSAR 7.0 can reduce LTS size by several orders of magnitude (e.g., 10^4), thus allowing larger LOTOS specifications to be handled.
- CÆSAR.BDD uses Binary Decision Diagrams to perform various structural analyses on basic Petri nets, such as exploring reachable markings to determine the set of "*dead*" transitions and the pairs of "*concurrent*" units.
- CÆSAR_SOLVE [11] is a generic software library for on-the-fly resolution of BESs, represented as implicit boolean graphs similarly to LTSs in OPEN/CÆSAR. It offers 6 resolution algorithms, based on breadth-first search or depth-first search (with memory-efficient variants for acyclic or disjunctive/conjunctive BESs) strategies, which also generate diagnostics (boolean subgraphs) explaining the truth value of boolean variables.
- DETERMINATOR [7] takes as input an extended CTMC encoded in the BCG format and tries to extract on-the-fly a pure CTMC (i.e., containing only stochastic transitions). Doing so, the tool checks a sufficient condition ensuring that the resulting CTMC is unique, or returns an error otherwise.
- DISTRIBUTOR [5] performs distributed LTS generation and on-the-fly reduction by τ-compression and τ-confluence [10] using several machines connected by a network. It launches a remote process on each machine to generate a fragment of the entire LTS. The result of the distributed generation is a PBG (*Partitioned* BCG *Graph*), i.e., a set of LTS fragments located on remote machines. DISTRIBUTOR is equipped with a graphical tool to monitor the PBG generation in real-time.
- EVALUATOR 3.5 [12, 11] evaluates on-the-fly, on an LTS, temporal properties expressed in regular alternation-free μ-calculus. The model checking problem is reformulated as a BES, which is solved using the CÆSAR_SOLVE library. The tool generates full diagnostics (examples and counterexamples) and optimizes memory consumption (i.e., does not store the transitions, but only the states of the LTS) for a large spectrum of properties. EVALUATOR 3.5 is 3 times faster and consumes 3 times less memory than EVALUATOR 3.0.

- Exp.Open 2.0 [8] maps communicating LTSs composed using synchronization vectors, parallel composition operators (from CCS, CSP, μCRL, LOTOS, and E-LOTOS), and/or generalized hide, rename, and cut operators onto the Open/Cæsar Api. It implements partial order reductions preserving various equivalences (e.g., branching, stochastic, weak trace equivalence). It can also translate communicating LTSs into the Pep, Tina, and Fc2 formats, and can synthesize interface constraints for refined compositional verification [9]. Exp.Open 2.0 uses 2 times less memory and is up to 45 times faster than Exp.Open 1.0.
- Projector 2.0 [13] reduces an LTS on-the-fly with respect to interface constraints represented by another LTS and a set of labels. Among its new features, Projector 2.0 allows to describe the set of labels more compactly by the way of regular expressions. Experiments indicate that Projector 2.0 is up to 4 times faster than Projector 1.0.
- Reductor 5.0 reduces an LTS on-the-fly, either partially or totally, modulo one out of 8 relations (trace, weak trace, $\tau^*.a$, and safety equivalences, τ-confluence, τ-compression, τ-divergence [10], and strong bisimulation), possibly displaying the equivalence classes of the quotient graph. Some of these reductions perform a local resolution of a BES using Cæsar_Solve. The algorithm used by Reductor 5.0 to eliminate invisible transitions has a lower average complexity than in earlier versions.
- Seq.Open [4] maps traces onto the Open/Cæsar Api. It implements an optimized representation of a trace as an LTS without storing the whole trace in memory, which is also significantly more efficient (up to 50 times faster) than converting the trace first into a BCG file.

Notice that the Aldébaran tool (available since the origin of Cadp) was replaced in Cadp 2006 by a (upward-compatible) shell script that invokes Bisimulator, Bcg_Info, Bcg_Min, and Reductor to provide the same functionalities as Aldébaran.

4 Conclusion

Cadp 2006 "*Edinburgh*" is the result of five years of intensive R&D in verification technology. Four computing platforms are currently supported: Sparc/Solaris, Intel/Linux, Intel/Windows, and PowerPC/Mac OS. As regards impact, 366 organizations have signed the Cadp license agreement already, and Cadp has been installed on 820 machines in the world during year 2006. Cadp was used for 94 case-studies[3] and 29 research tools[4] are connected to Cadp. Most notably, in the FormalFame project, Cadp was successfully used to validate crucial parts of Bull's NovaScale machines, which form the core of Tera10, Europe's most powerful supercomputer.

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

CADP will continue to be enhanced in the next years. In particular, we plan to apply CADP in three main areas: software environments for critical embedded systems (French projects OPENEMBEDD and TOPCASED), high-performance multiprocessor architectures (French project MULTIVAL), and bio-informatics (European project EC-MOAN, where CADP will contribute to the understanding of a bacterial model system).

Acknowledgements. Damien Bergamini, David Champelovier, Adrian Curic, Nicolas Descoubes, Holger Hermanns, Christophe Joubert, Bruno Ondet, Gordon Pace, Frédéric Perret, Irina Smarandache-Sturm, Gilles Stragier, Frédéric Tronel, and Marie Vidal contributed to CADP 2006. The authors are also grateful to the 79 CADP users[5] whose suggestions led to numerous CADP enhancements.

References

1. D. Bergamini, N. Descoubes, C. Joubert, and R. Mateescu. BISIMULATOR: A Modular Tool for On-the-Fly Equivalence Checking. In *Proc. of TACAS'2005*, vol. 3440 of LNCS, 2005.
2. J.-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu. CADP (CÆSAR/ALDEBARAN Development Package): A Protocol Validation and Verification Toolbox. In *Proc. of CAV'96*, vol. 1102 of LNCS, 1996.
3. H. Garavel, F. Lang, and R. Mateescu. An Overview of CADP 2001. *EASST Newsletter*, 4:13–24, 2002. Also available as INRIA Technical Report RT-0254.
4. H. Garavel and R. Mateescu. SEQ.OPEN: A Tool for Efficient Trace-Based Verification. In *Proc. of SPIN'2004*, vol. 2989 of LNCS, 2004.
5. H. Garavel, R. Mateescu, D. Bergamini, A. Curic, N. Descoubes, C. Joubert, I. Smarandache-Sturm, and G. Stragier. DISTRIBUTOR and BCG_MERGE: Tools for Distributed Explicit State Space Generation. In *Proc. of TACAS'2006*, vol. 2989 of LNCS, 2006.
6. H. Garavel and W. Serwe. State Space Reduction for Process Algebra Specifications. *Theoretical Computer Science*, 351(2):131–145, 2006.
7. H. Hermanns and C. Joubert. A Set of Performance and Dependability Analysis Components for CADP. In *Proc. of TACAS'2003*, vol. 2619 of LNCS, 2003.
8. F. Lang. EXP.OPEN 2.0: A Flexible Tool Integrating Partial Order, Compositional, and On-the-fly Verification Methods. In *Proc. of IFM'2005*, vol. 3771 of LNCS, 2005.
9. F. Lang. Refined Interfaces for Compositional Verification. In *Proc. of FORTE'2006*, vol. 4229 of LNCS, 2006.
10. R. Mateescu. On-the-fly State Space Reductions for Weak Equivalences. In *Proc. of FMICS'05*. ACM Computer Society Press, 2005.
11. R. Mateescu. CAESAR_SOLVE: A Generic Library for On-the-Fly Resolution of Alternation-Free Boolean Equation Systems. *Springer International Journal on Software Tools for Technology Transfer (STTT)*, 8(1):37–56, February 2006.
12. R. Mateescu and M. Sighireanu. Efficient On-the-Fly Model-Checking for Regular Alternation-Free Mu-Calculus. In *Science of Computer Programming*, 46(3):255–281, 2003.
13. G. Pace, F. Lang, and R. Mateescu. Calculating τ-Confluence Compositionally. In *Proc. of CAV'2003*, vol. 2725 of LNCS, 2003.

[5] listed at `http://www.inrialpes.fr/vasy/cadp/news6.html`