BISIMULATOR: A Modular Tool for On-the-Fly Equivalence Checking

Damien Bergamini, Nicolas Descoubes,
Christophe Joubert, and Radu Mateescu

INRIA Rhône-Alpes/VASY, 655, av. de l'Europe, 38330 Montbonnot St Martin, France
{Damien.Bergamini, Nicolas.Descoubes,
Christophe.Joubert, Radu.Mateescu}@inria.fr

1 Introduction

The equivalence checking problem consists in verifying that a system (e.g., a *protocol*) matches its abstract specification (e.g., a *service*) by comparing their Labeled Transition Systems (LTSs) modulo a given equivalence relation. Two approaches are traditionally used to perform equivalence checking: *global* verification requires to construct the two LTSs before comparison, whereas *local* (or *on-the-fly*) verification allows to explore them incrementally during comparison. The latter approach is able to detect errors even in prohibitively large systems, and therefore reveals more effective in combating state explosion.

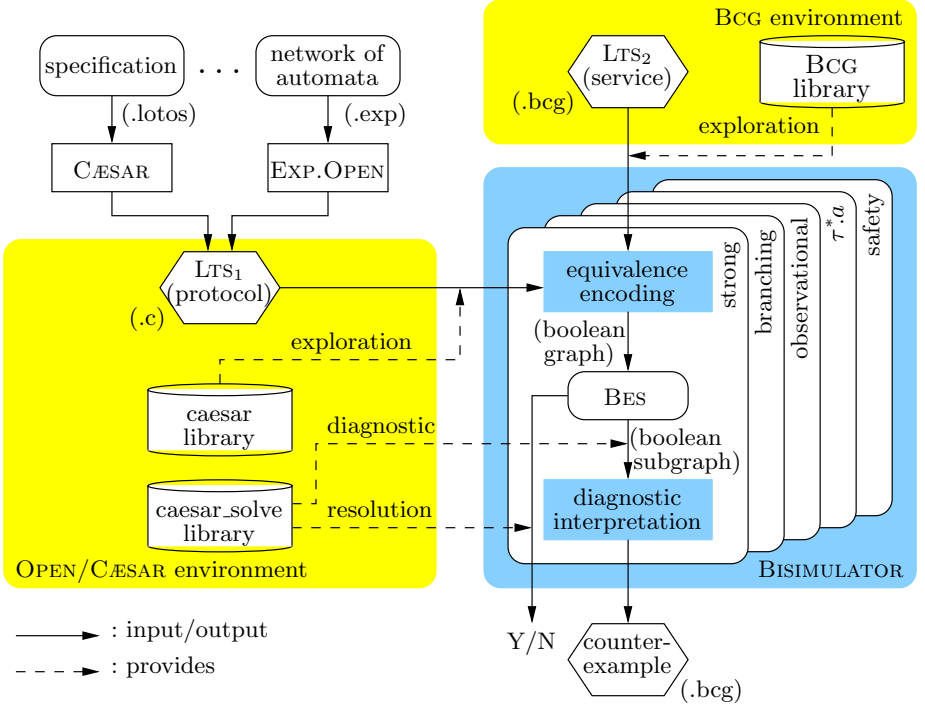
Existing on-the-fly equivalence checking algorithms (see [2] for a survey) explore the synchronous product of the two LTSs in a forward manner, until either a wrong execution pattern (counterexample) is encountered, or the product is entirely explored (the LTSs are equivalent). Despite their usefulness, only a few implementations of these algorithms are available, most of them being targeted to specific input languages and/or equivalence relations. This is the case for ALDÉBARAN [4], whose efficient on-the-fly algorithms [3] only handle networks of communicating automata, being difficult to adapt to other description languages, such as process algebras. In this context, a more generic technology is desirable in order to reduce the development effort, handle new equivalence relations easily, and achieve a maximal reuse of existing algorithms.

In this paper, we present BISIMULATOR, an efficient on-the-fly equivalence checker with a highly modular architecture, developed within the CADP verification toolbox [6]. The front-end of the tool encodes five widely-used equivalence relations in terms of Boolean Equation Systems (BESs) by using the OPEN/CÆSAR [5] and BCG environments of CADP, which provide powerful LTS exploration primitives. This makes BISIMULATOR language-independent, the tool being directly available for any description language equipped with a compiler able to produce LTSs compliant with the OPEN/CÆSAR interface. The back-end of the tool carries out the verification by means of the generic CÆSAR_SOLVE [9] library of CADP, dedicated to (both sequential and distributed) on-the-fly BES resolution and diagnostic generation. This architecture clearly separates the implementation of equivalence relations and the verification engine, which can therefore be extended and optimized independently.

N. Halbwachs and L. Zuck (Eds.): TACAS 2005, LNCS 3440, pp. 581–585, 2005.

2 Tool Architecture

Bisimulator (see below) takes as inputs two Ltss $\langle Q_i, A_i, T_i, q_{0_i} \rangle$ $(i \in \{1,2\})$, where Q_i are the sets of states, A_i the sets of actions, $T_i \subseteq Q_i \times A_i \times Q_i$ the transition relations, and $q_{0_i} \in Q_i$ the initial states. The first Lts is represented implicitly (by its successor function) as an Open/Cæsar program obtained by translating a system description, and the second one is represented explicitly (by its list of transitions) as a Bcg file[1]. Bisimulator (12,000 lines of C code) consists of several modules, each one containing the Bes translation and the diagnostic generation for a particular equivalence relation (strong, branching, observational, $\tau^*.a$, safety). Bess are derived directly from the definitions of equivalence relations; for instance, strong equivalence is translated into the greatest fixed point Bes $\{X_{p,q} \overset{\nu}{=} \bigwedge_{p \xrightarrow{a} p'} \bigvee_{q \xrightarrow{a} q'} X_{p',q'} \wedge \bigwedge_{q \xrightarrow{a} q'} \bigvee_{p \xrightarrow{a} p'} X_{p',q'}\}$, where each variable $X_{p,q}$ is true iff the states $p \in Q_1$ and $q \in Q_2$ are strongly equivalent.



Bess are handled internally by the Cæsar_Solve library as *boolean graphs* [1], which give an intuitive view of the dependencies between variables and facilitate the development of resolution algorithms. Boolean graphs are represented implicitly by their successor function, in the same way as Ltss

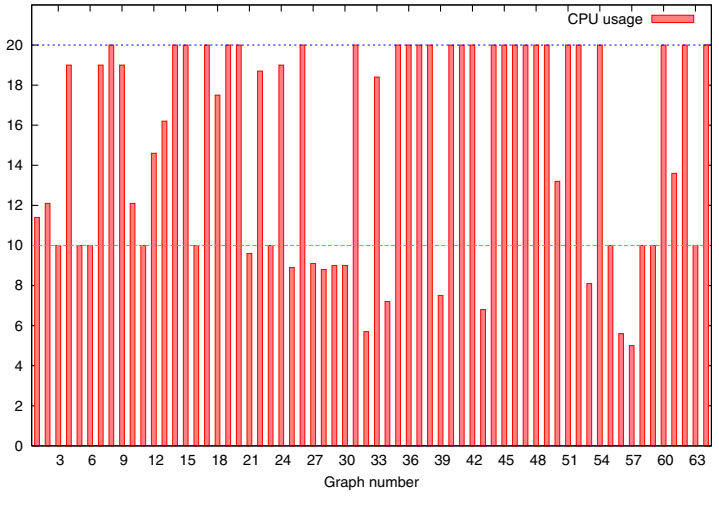
[1] This asymmetry, due to the current architecture of Open/Cæsar, which does not allow to explore several Ltss on-the-fly, is likely to disappear in a future version.

in OPEN/CÆSAR. The library offers several on-the-fly resolution algorithms, based on different search strategies of boolean graphs: breadth-first, which produces small-depth diagnostics, and depth-first, with memory-efficient variants for acyclic or disjunctive/conjunctive boolean graphs (these kinds of graphs are obtained, e.g., by encoding comparison modulo strong equivalence when one LTS is acyclic or deterministic, respectively) [9]. Diagnostics are provided by the library as boolean subgraphs, which are subsequently converted by BISIMULATOR into counterexamples (directed acyclic graphs containing transition sequences that can be executed simultaneously in the two LTSs and lead to non equivalent states) represented as BCG files.

Recently, CÆSAR_SOLVE has been extended with a distributed on-the-fly resolution algorithm [8] running on several machines connected by a network. This allowed to immediately obtain a distributed version of BISIMULATOR, which scales up smoothly to larger systems.

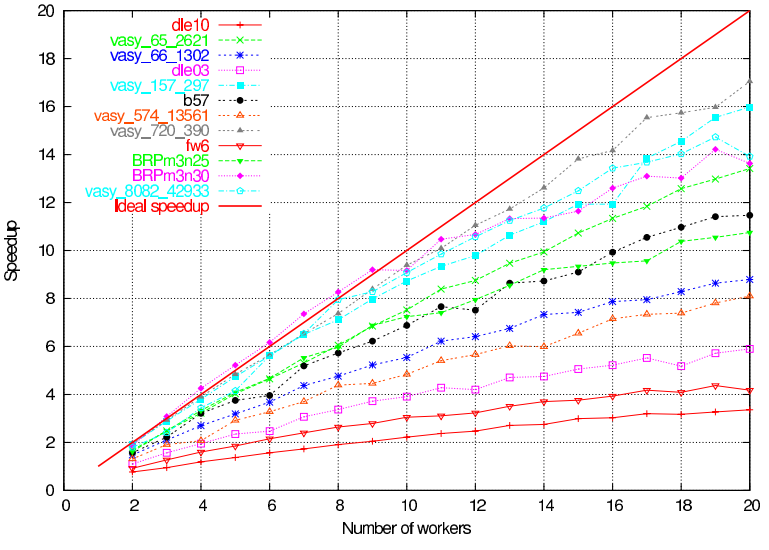
3 Performance Measures

We give below some experimental data obtained using various LTSs taken from the CADP demo examples and from the VLTS benchmark suite [10].



The first picture shows a comparison between BISIMULATOR and ALDÉBARAN (on-the-fly algorithms) for strong equivalence, based on experiments performed using 64 LTSs ranging from 3 Kstates and 6 Ktransitions to 3.8 Mstates and 11 Mtransitions, on a PC with 2.2 GHz and 1 Gbyte of memory. Each experiment consisted in checking that an LTS is equivalent with its minimized version modulo strong equivalence, which is a worst-case situation for on-the-fly algorithms, since both LTSs must be explored entirely. Each vertical line on the picture denotes a mark (between 0 and 20) comparing the speed of the tools

on a given experiment. The mark is computed as follows: 20 if BISIMULATOR succeeds and ALDÉBARAN fails; 19 if BISIMULATOR is more than 5 times faster; 10..19 if BISIMULATOR is from 1 to 5 times faster; 10 if both tools are equally fast or they fail; 0..10 in a strictly symmetric way when BISIMULATOR is slower or fails. On 31 experiments out of 64, ALDÉBARAN fails because of memory shortage or too long computation, whereas BISIMULATOR only fails (together with ALDÉBARAN) on 4 experiments. On the remaining 33 experiments, the average time/memory are 11.8 sec./32.5 Mbytes for BISIMULATOR, and 20.5 sec./99 Mbytes for ALDÉBARAN.



The second picture shows the speedup of the distributed version of BISIMULATOR w.r.t. the sequential one (breadth-first search algorithm) for strong equivalence, based on experiments performed using 12 LTSs ranging from 65 Kstates and 2.6 Mtransitions to 8 Mstates and 42.9 Mtransitions, on a PC cluster composed of 20 nodes with 2.4 GHz and 1.5 Gbytes of memory. Each experiment consisted in comparing an LTS with its minimized version. Speedup ranges uniformly from low – still better than sequential – to almost optimal, and increases with LTS size (e.g., the experiment `vasy_157_297`, involving an LTS with 157 Kstates and 297 Ktransitions, is handled 16 times faster than the sequential version by the distributed version using 20 machines). Similar behaviours are observed for weak equivalences; additional experimental data showing low memory overhead and good scalability of distributed BISIMULATOR is available in [8].

4 Conclusion and Future Work

The development of an on-the-fly equivalence checker "from scratch" is a complex and costly task. The modular architecture adopted for BISIMULATOR aims

at making this process easier, by using the well-established verification framework of BESS, together with the generic libraries for LTS exploration and BES resolution provided by CADP. This tool architecture reduces the effort of implementing a new equivalence relation to its strict minimum: encoding the mathematical definition of the equivalence as a BES, and interpreting the counterexamples. Another advantage of our approach over previous dedicated on-the-fly equivalence checking algorithms [3] is that particular cases suitable for optimization can be handled more elegantly and precisely using the BES representation. For instance, in BISIMULATOR the encodings of equivalence relations exploit the determinism w.r.t. a given action and the absence of τ-transitions locally (i.e., on each state encountered during verification) to reduce the size of boolean equations, whereas in [3] the condition for applying the optimized algorithm handling the "deterministic case" is global (i.e., it involves all states of one LTS).

We plan to continue our work by extending BISIMULATOR with other equivalence relations (e.g., trace equivalence and its weak variant, Markovian bisimulation [7], etc.) and by studying new strategies for (sequential and distributed) on-the-fly BES resolution.

References

1. H. R. Andersen. Model Checking and Boolean Graphs. *Theoretical Computer Science*, 126(1):3–30, April 1994.
2. R. Cleaveland and O. Sokolsky. *Equivalence and Preorder Checking for Finite-State Systems.* In J. A. Bergstra, A. Ponse, and S. A. Smolka (eds.), *Handbook of Process Algebra*, chapter 6, pages 391–424. North-Holland, 2001.
3. J-C. Fernandez and L. Mounier. Verifying Bisimulations "On the Fly". In J. Quemada, J. Manas, and E. Vázquez (eds.), *Proc. of FORTE'90 (Madrid, Spain).* North-Holland, November 1990.
4. J-C. Fernandez and L. Mounier. A Tool Set for Deciding Behavioral Equivalences. In *Proc. of CONCUR'91 (Amsterdam, The Netherlands)*, August 1991.
5. H. Garavel. OPEN/CÆSAR: An Open Software Architecture for Verification, Simulation, and Testing. In B. Steffen (ed.), *Proc. of TACAS'98 (Lisbon, Portugal)*, LNCS vol. 1384, pp. 68–84. Springer Verlag, March 1998. Full version available as INRIA Research Report RR-3352.
6. H. Garavel, F. Lang, and R. Mateescu. An Overview of CADP 2001. *EASST Newsletter* 4:13–24, August 2002. Also available as INRIA Report RT-0254.
7. H. Hermanns and M. Siegle. Bisimulation Algorithms for Stochastic Process Algebras and their BDD-based Implementation. In J-P. Katoen (ed.), *Proc. of ARTS'99 (Bamberg, Germany)*, LNCS vol. 1601, pp. 244–265. Springer Verlag, May 1999.
8. C. Joubert and R. Mateescu. Distributed On-the-Fly Equivalence Checking. In L. Brim and M. Leucker (eds.), *Proc. of PDMC'04 (London, United Kingdom)*, ENTCS, September 2004. To appear.
9. R. Mateescu. A Generic On-the-Fly Solver for Alternation-Free Boolean Equation Systems. In H. Garavel and J. Hatcliff (eds.), *Proc. of TACAS'2003 (Warsaw, Poland)*, LNCS vol. 2619, pp. 81–96. Springer Verlag, April 2003. Full version available as INRIA Research Report RR-4711.
10. `http://www.inrialpes.fr/vasy/cadp/resources/benchmark_bcg.html`