A Set of Performance and Dependability Analysis Components for CADP

Holger Hermanns[1,2] and Christophe Joubert[3]

[1] Department of Computer Science, University of Twente,
P.O. Box 217, NL-7500 AE Enschede, The Netherlands
[2] Department of Computer Science, Saarland University,
Im Stadtwald, D-66123 Saarbrücken, Germany
[3] INRIA Rhône-Alpes / VASY, 655, avenue de l'Europe
F-38330 Montbonnot Saint-Martin, France

Abstract. This paper describes a set of analysis components that open the way to perform performance and dependability analysis with the CADP toolbox, originally designed for verifying the functional correctness of LOTOS specifications. Three new tools (named BCG_STEADY, BCG_TRANSIENT and DETERMINATOR) have been added to the toolbox. The approach taken fits well within the existing architecture of CADP which doesn't need to be altered to enable performance evaluation.

1 Introduction

The design of models suited for performance and dependability analysis of systems is difficult because of their ever increasing size and complexity, in particular for systems with a high degree of irregularity. The potential of formal methods and tools to support the modelling and analysis of performance and dependability aspects has led to various techniques and tools, mostly based on stochastic Petri nets (SPN for short, e.g. [2, 22, 3, 4]), or stochastic process algebras (e.g. [12, 1, 15]), or both [7].

This paper describes PDAC (Performance and Dependability Analysis Components), a set of components that enable the study of performance and dependability for specifications developed by means of the CADP toolbox [10]. The latter is a widespread tool set for the design and verification of complex systems. CADP supports the process algebra LOTOS for specification, and offers various tools for simulation and formal verification, including equivalence checkers (bisimulations) and model checkers (temporal logics and modal μ-calculus). The toolbox is designed as an open platform for the integration of other specification, verification and analysis techniques. This is realized by means of application programming interfaces (API) which on different levels provide means to extend or exploit the functionalities of the toolbox. These APIs have been used by others to link CADP to other specification languages as well as other verification/testing tools.

Here we describe how these APIs have been used to support performance and dependability analysis based on Markov modelling and numerical algorithms. Our efforts have been driven by the intention to avoid changes to the

existing components as much as possible, while providing a sound and efficient framework for performance and dependability analysis, including state-of-the-art stochastic model checking techniques. To achieve this we use the theory of interactive Markov chains [13], a conservative extension of both process algebra and *continuous-time Markov chains* (CTMC for short), the latter being a well-investigated and frequently used class of stochastic models. More details on the modelling philosophy can be found in [13, 8], while here we focus on the tool architectural aspects. The resulting set of tool components (`http://fmt.cs.utwente.nl/tools/pdac/`) will be part of the forthcoming CADP 2003 (`http://www.inrialpes.fr/vasy/cadp`).

The paper is organised as follows. Section 2 briefly explains how the process algebra LOTOS can be used for modelling Markovian aspects. Section 3 describes extensions of CADP to support performance evaluation.

2 Interactive Markov chains in CADP

Many stochastic models derived from state-transition diagrams have been proposed. Our approach is based on the *interactive Markov chain* model (IMC), which can be considered as simply a *labelled transition system* (LTS) whose transitions can be either labelled with an action (as in an 'ordinary' LTS) or with special labels of the form "`rate` λ", where λ is a positive real value. A transition "$\xrightarrow{\texttt{rate}\ \lambda}$" going out of some state S is called a *delay transition* and expresses an internal delay in state S (henceforth called a Markov delay). It indicates that the time t spent in S follows a so-called *negative exponential distribution function* $Prob\{t \leq x\} = 1 - e^{-\lambda x}$, to be read as: the probability that state S is exited at time x the latest equals $1 - e^{-\lambda x}$. The IMC model contains as two particular cases the LTS model and the well-known CTMC model (which is obtained when there are only delay transitions). The latter model has been extensively studied in the literature and is equipped with various efficient evaluation strategies (see, e.g. [23]). Similar to Markov decision processes [20], the IMC model allows nondeterminism in states, i.e., two identical action transitions leaving the same state.

To extend the CADP tool towards IMC, the approach chosen [8] is a lightweight one, which does not modify the syntax of LOTOS and requires no change in the CADP compilers for LOTOS (CÆSAR.ADT and CÆSAR). The approach has two steps. Starting from a (functionally verified) LOTOS specification, the user can insert, wherever a Markov delay λ_i should occur, a new (fresh) LOTOS gate Λ_i. After the special gates Λ_i have been inserted in the specification, CÆSAR and CÆSAR.ADT are invoked as usual to generate the corresponding LTS, which is stored in the BCG (Binary Coded Graphs) format. This LTS is then turned into an IMC (still encoded in the BCG format) by replacing all its action transitions Λ_i with delay transitions "`rate` λ_i". This is done using the BCG_LABELS tool of CADP.

So, by this two step methodology, we can generate an IMC model corresponding to a LOTOS specification with inserted delays and store this model in the

BCG format. Another, less manual, possibility to specify IMC has been suggested in [16] where a constraint-oriented style is used to incorporate Markov delays (or more complex phase-type distributions) between the actions of an existing (and verified) LOTOS specification. Again, the result is an IMC model stored in the BCG format. We refer to [8] for a discussion of the soundness of this approach, and for more details and options in the process of generating an IMC with CADP.

3 Analysis components

The PDAC set encompasses two sorts of tools to analyse IMC models generated via CADP. (i) The DETERMINATOR tool and the CADP component tool BCG_MIN provide different means to distill a CTMC from an IMC model. (ii) The main analysis components, BCG_STEADY and BCG_TRANSIENT, enable numerical inspection of the behavior CTMCs encoded in the BCG format. We discuss the tools in reverse order.

Analysing a Markov Chain. Two analysis tools provide standard numerical algorithms to compute the distribution of probability in a CTMC.

- BCG_TRANSIENT implements the uniformisation method [23], calculating the time-dependent probability to be in each of the Markov Chain states at a user-specified point in time (relative to the initialisation of the system). The time point is a command-line parameter.
- BCG_STEADY computes the time-independent, long run equilibrium probabilities for each of the system states, using the Gauss-Seidel algorithm [23]. This equilibrium is known to exist for arbitrary finite CTMCs.

Both tools accept a BCG file as input. Unless the file contains action transitions, the graph is accepted for analysis. First, the tool transforms the given graph into the generator matrix representation of a CTMC using the sparse matrix package of [18]. Then, the respective computational procedure is launched. In either case, this results in a vector of state probabilities.

Dependent on the option selected by the user, the solution vector is written to file (option `-sol`) or is further processed, to compute so-called transition throughputs. A transition throughput indicates the average number of transition executions per time unit, for a user-specified set of transitions of interest. These measures can provide important high-level information to assess the system performance, reliability or productivity. BCG_TRANSIENT and BCG_STEADY support throughput calculations if the BCG file contains tagged delay transitions of the form "*tag* `; rate` λ", where *tag* can be an arbitrary label. In this case the throughput is computed for each syntactic tag occuring in the BCG file (if the option `-thr` is selected).

To allow postprocessing and visualisation of computed measures, BCG_TRANSIENT and BCG_STEADY give output results in a CSV-like format (Comma Separated Values). Thus data can be directly conveyed as input to table-oriented applications such as GNUPLOT or EXCEL, which can read CSV files.

Distilling the Markov Chain. Since the numerical analysis components take a CTMC as input, tools are needed to distill a CTMC from an IMC model. Due to the presence of nondeterminism in IMC this is infeasible in general, and the toolset only provides two partial solutions to this. Both are based on the observation that nondeterminism – while being essential for compositional specification – can often be factored out in the final state space being subject to performance and dependability analysis.

- DETERMINATOR implements an on-the-fly algorithm for the *well-specified* condition [11, 6] of a stochastic process. Roughly, an IMC is said to be well-specified, if – whatever nondeterministic decisions are taken – the resulting CTMC is unique. The algorithm implemented is a variant of the one described in [5, 6].
- BCG_MIN is the bisimulation minimiser of CADP. It can also be used for distilling a CTMC from an IMC: If for a given IMC it holds that – whatever nondeterministic decisions are taken – the resulting *lumped* CTMC chain is unique, then this lumped CTMC will be returned by (the stochastic branching bisimulation option of) BCG_MIN applied to the original IMC model with all actions hidden.

BCG_MIN provides a more powerful way of resolving nondeterministic compared to the DETERMINATOR tool (which does not check the quotient under lumping), but it is computationally more expensive (and is not an on-the-fly algorithm). It is possible to use DETERMINATOR as a preprocessor to BCG_MIN. In the practical cases we considered, this combination turned out to be rather beneficial. Typically the time needed for distilling a lumped CTMC from IMC was decreased by a factor of eight, compared to applying BCG_MIN directly.

Other specification formalisms. When designing the extension to CADP, care has been taken to exploit the features of the BCG API in a way that as far as possible also other formalisms are supported by the toolset. We refer to the manual pages (see `http://fmt.cs.utwente.nl/tools/pdac/` and `http://www.inrialpes.fr/vasy/cadp/man`) for a complete description of the possibilities and limitations, and highlight only a few specific options we have implemented:

- Both BCG_TRANSIENT and BCG_STEADY can handle state spaces where delay transitions and probabilistic transitions coexist. The latter are encoded using distinguished labels of the form "`prob` p_i", where p_i is a real value from the interval $(0, 1]$. These models often appear in the context of generalized SPN and similar models [19].
- DETERMINATOR is able to handle models with delay transitions, nondeterministic transitions and also probabilistic transitions. Such models occur in the context of stochastic activity networks or other SPN like models [21, 5].

These options allow the tool components to be used in the context of other modelling formalisms, provided that a link to the BCG format exists (via the API). The tool BCG_IO provides encoding and decoding functionality from/to

various tool formats, including simple text-based formats. As one particular application of the BCG API, a conversion from the BCG format to the ETMCC model checker [17] has been implemented. This allows one to perform stochastic model checking of CTMCs encoded in the BCG format.

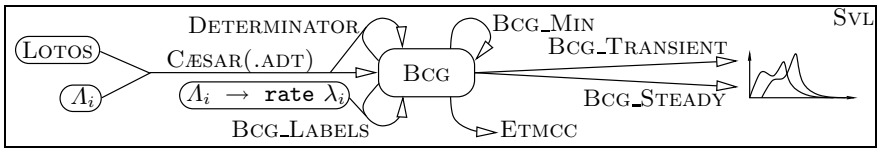

Fig. 1. Tool chain of CADP with PDAC.

Using the scripting language SVL. The diagram in Figure 1 summarizes the position of each component inside the tool chain, where Λ_i refers to the fresh LOTOS gates which are later (using BCG_LABELS) replaced by "`rate` λ_i". The tools BCG_MIN and/or DETERMINATOR are used to distill a CTMC (the latter being an *on-the-fly* algorithm), and BCG_STEADY or BCG_TRANSIENT are used for numerical analysis.

To drive a whole set of performance studies, the scripting language SVL[9] turned out to be very useful. It allows one to conveniently execute the required sequence of tool invocations, and to iterate the numerical analysis on a (possibly multidimensional) list of performance parameters. With SVL we have mechanised the analysis of the SCSI II protocol [8] and of the HUBBLE space telescope [14].

Acknowledgements. We are grateful to Hubert Garavel and Radu Mateescu (INRIA Rhône-Alpes) for their insightful comments about the design of the PDAC. The BCG_MIN tool is a co-development of Damien Bergamini (INRIA Rhône-Alpes), Moëz Cherif (formerly at INRIA Rhône-Alpes), Hubert Garavel, and Holger Hermanns. The tools BCG_TRANSIENT and BCG_STEADY are based on code provided by Vassilis Mertsiotakis (LUCENT TECHNOLOGIES) as part of the TIPPTOOL, which was developed at the University of Erlangen-Nürnberg. This work is supported by the Dutch foundation for scientific research NWO under the VERNIEUWINGSIMPULS program, and has been carried out during a stay of the second author at the University of Twente.

References

1. M. Bernardo, W.R. Cleaveland, S.T. Sims, and W.J. Stewart. TwoTowers: A Tool Integrating Functional and Performance Analysis of Concurrent Systems. In *Proc. FORTE/PSTV'98*, p. 457–467, Kluwer, 1998.
2. G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN 1.7: GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets. *Perf. Eval.*, 24(1,2):47–68, 1995.

3. G. Ciardo, A.S. Miner. SMART: Simulation and Markovian Analyzer for Reliability and Timing. In *Proc. IPDS'96*, p. 60, IEEE CS Press, 1996.
4. G. Ciardo, J. Muppala, and K. Trivedi. SPNP: stochastic Petri net package. In *Proc. PNPM'89*, p. 142–151, IEEE CS Press, 1989.
5. G. Ciardo and R. Zijal. Well-defined stochastic Petri nets. In *Proc. MASCOTS'96*, p. 278–284, IEEE CS Press, 1996.
6. D.D. Deavours and W.H. Sanders. An efficient well-specified check. In *Proc. PNPM'99*, p. 124–133, IEEE CS Press, 1999.
7. D.D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J.M. Doyle, W. H. Sanders, and P.G. Webster. The Möbius Framework and Its Implementation. *IEEE Trans. on Softw. Eng.*, 28(10):956–969, 2002.
8. H. Garavel and H. Hermanns. On Combining Functional Verification and Performance Evaluation using CADP. In *Proc. FME'02*. LNCS 2391:410–429, 2002.
9. H. Garavel and F. Lang. SVL: A Scripting Language for Compositional Verification. In *Proc. FORTE/PSTV 2001*, p. 377–394, Kluwer, 2001.
10. H. Garavel, F. Lang, and R. Mateescu. An Overview of CADP 2001. *EASST Newsletter*, 4:13–24, 2002.
11. R. German, A. van Moorsel, M.A. Qureshi, and W.H. Sanders. Algorithms for the Generation of State-Level Representations of Stochastic Activity Networks with General Reward Structures. *IEEE Trans. on Softw. Eng.*, 22(9):603–614, 1996.
12. S. Gilmore and J. Hillston. The PEPA Workbench: A Tool to Support a Process Algebra-Based Approach to Performance Modelling. In *Proc. TOOLS'94*. LNCS 794:353–368, 1994.
13. H. Hermanns. *Interactive Markov Chains and the Quest for Quantified Quality.* LNCS 2428, 2002.
14. H. Hermanns. Construction and Verification of Performance and Reliability Models. *Bulletin of the EATCS*, 74:135-154, 2001.
15. H. Hermanns, U. Herzog, U. Klehmet, V. Mertsiotakis, and M. Siegle. Compositional performance modelling with the TIPPtool. *Perf. Eval.*, 39(1-4):5–35, January 2000.
16. H. Hermanns and J.P. Katoen. Automated compositional Markov chain generation for a plain-old telephony system. *Sci. of Comp. Prog.*, 36(1):97–127, 2000.
17. H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. A Markov Chain Model Checker. In *Proc. TACAS'00*. LNCS 1785:347–362, 2000.
18. K.S. Kundert. Sparse matrix techniques. In *Circuit analysis, Simulation and Design 3*, North-Holland, 1986.
19. A. Marsan, G. Balbo, and G. Conte. A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM Trans. on Comp. Sys.*, 2(2):93–122, 1984.
20. M.L. Puterman. *Markov Decision Processes.* John Wiley, 1994.
21. W.H. Sanders and J.F. Meyer. Stochastic Activity Networks: Formal Definitions and Concepts. In *Proc. FMPA 2000*. LNCS 2090:315–343, 2001.
22. W.H. Sanders, W.D. Oball, M.A. Qureshi, and F.K. Widjanarko. The UltraSAN modeling environment. *Perf. Eval.*, 24(1):89–115, 1995.
23. W.J. Stewart. *Introduction to the numerical solution of Markov chains.* Princeton University Press, 1994.