A Markov Chain Model Checker*

Holger Hermanns[a†], Joost-Pieter Katoen[a],
Joachim Meyer-Kayser[b‡], and Markus Siegle[b]

[a] *Formal Methods and Tools Group, University of Twente*
P.O. Box 217, 7500 AE Enschede, The Netherlands

[b] *Lehrstuhl für Informatik 7, University of Erlangen-Nürnberg*
Martensstraße 3, 91058 Erlangen, Germany

Abstract

Markov chains are widely used in the context of performance and reliability evaluation of systems of various nature. Model checking of such chains with respect to a given (branching) temporal logic formula has been proposed for both the discrete [6] and the continuous time setting [1, 3]. In this note, we describe the prototype model checker $\mathsf{E} \vdash \mathsf{MC}^2$ for discrete and continuous-time Markov chains, where properties are expressed in appropriate extensions of CTL. We illustrate the general benefits of this approach and discuss the structure of the tool.

Model checking Markov chains

Markov chains are widely used as adequate models in many diverse areas, ranging from mathematics and computer science to other disciplines such as operations research, industrial engineering, biology and demographics. Markov chains can be used to estimate performance characteristics of various nature, for instance to quantify throughput of manufacturing systems, locate bottlenecks in communication systems, or to estimate reliability in aerospace systems.

Model checking is a very successful technique to establish the *correctness* of systems from similar application domains, usually described in terms of a non-deterministic finite-state model. If non-determinism is replaced by randomized, i.e. probabilistic decisions, the resulting model boils down to a finite-state

*supported by the NWO-DFG bilateral cooperation program (VOSS).
†supported by the Netherlands Organisation for Scientific Research (NWO).
‡supported by the German Research Council DFG under HE 1408/6-1.

discrete-time Markov chain (DTMC). For these models, a number of qualitative and quantitative model checking algorithms have been proposed. In a qualitative setting it is checked whether a property holds with probability 0 or 1; in a quantitative setting it is verified whether the probability for a certain property meets a given lower or upper bound. PCTL [6] is a representative of the latter kind. It is an extension of CTL [5], allowing one to specify and verify properties such as *"After a system failure, the probability that the system will not come up again is at most* 10^{-6}*."*

Markov chains are *memoryless*. In the discrete-time setting this is reflected by the fact that probabilistic decisions do not depend on the outcome of decisions taken earlier, only the state currently occupied is decisive to completely determine the probability of next transitions. For continuous-time Markov chains (CTMCs), where time ranges over (positive) reals (instead of discrete subsets thereof) the memoryless property further implies that the probabilities of taking next transitions do not depend on the amount of time spent in the current state. DTMCs are mostly applied to strictly synchronous scenarios, while CTMCs have shown to fit in well with (interleaving) asynchronous scenarios. In particular, CTMCs are the underlying semantic model of major high-level performance modelling formalisms such as stochastic Petri nets, stochastic automata networks, stochastic process algebras, Markovian queueing networks, and various extensions thereof.

Recently, the logic CSL has been proposed [1, 3], an extension of both PCTL and CTL tailored to quantitative properties of CTMCs. Apart from CTL and PCTL properties, a selection of typical properties that can be verified using this logic is:

- *"After a system failure, there is at least a 99.99 % chance that the system will come up again within 5 time units."*
- *"On the long run, the probability of the system being unavailable is at most* 10^{-4}*."*
- *"The probability that signal 'ready' will be received within the next* 4 *time units is more than* 0.3*."*

In this short paper we describe the *Erlangen–Twente Markov Chain Checker* ($\mathsf{E} \vdash \mathsf{MC}^2$), to our knowledge the first implementation of a model checker for DTMCs and CTMCs. It uses numerical methods to model check PCTL and CSL-formulas, based on [6, 3, 2]. Apart from standard graph algorithms, model checking CSL involves matrix-vector multiplication, solution of linear systems of equations, and solution of systems of Volterra integral equations. Linear systems of equations are solved iteratively by standard numerical methods [15]. Two alternatives to solve systems of integral equations are implemented: One is based on piecewise integration of discretized distribution functions, the other is based on uniformisation [2, 14]. Uniformisation is the default option, because it allows

the tool to a priori calculate the computational effort needed to check a given property. This effort depends on the numerical parameters of the current model, on the property to be checked, and on the required numerical precision ε (the latter is a parameter set by the user).

$\mathsf{E} \vdash \mathsf{MC}^2$ is a *global* model checker, i.e. it checks the validity of a formula for all states in the model. It has been developed such that it can easily be linked to a wide range of existing high-level modelling tools based on, for instance, stochastic process algebras, stochastic Petri nets, or queueing networks. A whole variety of such tools exists [7], most of them using dedicated formats to store the transition matrix $\mathbf{R}$ of the Markov chain that is obtained from a high-level specification. This matrix encodes the probabilistic behaviour of the system as time passes. Together with a labelling function L, which associates the states of the Markov chain with sets of atomic propositions, the matrix $\mathbf{R}$ constitutes the interface between the high-level formalism at hand and the model checker. Currently, the tool accepts DTMCs and CTMCs represented in a format generated by the stochastic process algebra tool TIPPtool [13], but the tool is designed in such a way that it can easily bridge to various other input formats. The stochastic Petri net tool DaNAMiCS [4] has recently been extended to generate input for $\mathsf{E} \vdash \mathsf{MC}^2$.

Tool architecture

The tool has been written entirely in Java (version 1.2), in order to provide platform independence and to enable fast and efficient program development. Furthermore, support for the development of graphical user interfaces as well as grammar parsers is at hand. For the sake of simplicity, flexibility and extensibility we abstained from low-level optimizations, such as minimization of object invocations. The design and implementation took approximately 15 man-months, with about 10000 lines of code for the kernel and 1500 lines of code for the GUI implementation, using the Swing library. The tool architecture consists of five components:

Graphical User Interface (cf. Fig. 1) enables the user to load, modify and save verification projects. Each project consists of a model $\mathbf{R}$, a labelling L, and the properties to be checked. The GUI contains the 'CSL Property Manager' which allows the user to construct and edit CSL-formulas. The GUI also prints results and additional logging information on screen or writes them into file. Several verification parameters for the numerical analysis, such as solution method, precision ε, and number of interpolation points for the piecewise integration, can be set by the user.

Tool Driver controls the model checking procedure. It generates the parse tree corresponding to a given CSL property. Subsequent evaluation of the parse tree issues calls to the respective verification objects that encapsulate the

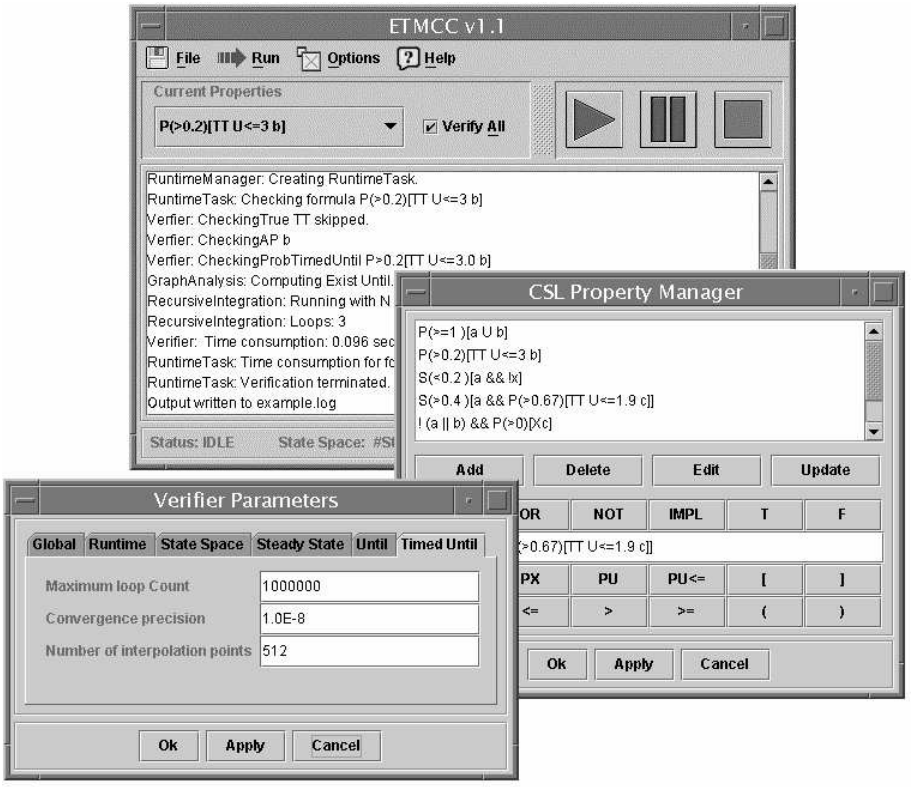

Figure 1: User interface of $\mathsf{E} \vdash \mathsf{MC}^2$

verification sub-algorithms. These objects, in turn, use the analysis and/or numerical engine.

Analysis Engine is the engine that supports standard model checking algorithms for CTL-style until-formulas, as well as graph algorithms, for instance to compute the bottom strongly connected components of a Markov chain. The former algorithms are very useful in a pre-processing phase during the checking of probabilistic until-formulas (they may help to avoid many numerical calculations), while the latter is needed when calculating long-run average properties.

Numerical Engine is the numerical analysis engine of the tool. It provides several methods for the numerical solution of linear systems, for numerical integration, and for uniformisation. These are used to solve sytems of linear or integral equations on the basis of parameters provided by the user via the GUI.

State Space Manager represents DTMCs and CTMCs in a uniform way. In

fact, it provides an interface between the various checking and analysis components and the way in which DTMCs and CTMCs are actually represented. This eases the use of different, possibly even symbolic (i.e. BDD-based) state space representations. It is designed to support input formats of various kinds, by means of a simple plug-in-functionality (using JAVA's dynamic class loading capability). It maintains information about the validity of atomic propositions and of sub-formulas for each state, encapsulated in a 'Sat' sub-component. After checking a sub-formula, this sub-component stores the results, to be used later. In the current version of the tool, the state space is represented as a sparse matrix [15]. All real values are stored in the IEEE 754 floating point format with double precision (64 bit).

Case studies

Even though the tool is still a prototype, it has already been used in a number of nontrivial case studies, including

- validation and performance assessment of a cyclic server polling system [8],
- reliability estimation of the Hubble space telescope [9],
- dependability analysis of a workstationclusters [10], and
- performance and availability analysis of a distributed database server [11].

Conclusion

This short paper describes the Markov chain model checker $\mathsf{E} \vdash \mathsf{MC}^2$. For more information about the tool, the reader is invited to consult [8], or `http://www7.informatik.uni-erlangen.de/etmcc/`. For academic purposes the tool can be downloaded free of charge via this web-site. The tool is currently being extended towards the model checking of action-oriented properties expressed in the logic aCSL [11, 12].

References

[1] A. Aziz, K. Sanwal, V. Singhal and R. Brayton. Verifying continuous time Markov chains. In *Computer Aided Verification, CAV 96*, Springer LNCS 1102: 269–276, 1996.

[2] C. Baier, B.R. Haverkort, H. Hermanns and J.-P. Katoen. Model checking continuous-time Markov chains by transient analysis. In *Computer Aided Verification, CAV 2000*, Springer LNCS 1855: 358-372, 2000.

[3] C. Baier, J.-P. Katoen and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In *CONCUR 99*, Springer LNCS 1664: 146–162, 1999.

[4] B. Changuion, I. Davies, and M. Nelte. DaNAMiCS - a Petri Net Editor. `http://www.cs.uct.ac.za/Research/DNA/DaNAMiCS/`.

[5] E.M. Clarke, E.A. Emerson and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Tr. on Progr. Lang. and Sys.*, **8**(2): 244-263, 1986.

[6] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Form. Asp. of Comp.*, **6**(5): 512–535, 1994.

[7] B.R. Haverkort and I.G. Niemegeers. Performability modelling tools and techniques. *Performance Evaluation* **25**: 17–40, 1996.

[8] H. Hermanns, J.P. Katoen, J. Meyer-Kayser and M. Siegle. A Markov chain model checker. In *TACAS 2000*, Springer LNCS 1785: 347–362, 2000.

[9] H. Hermanns. Performance and reliability model checking and model construction. In *Formal Methods for Industrial Critical Systems, FMICS 2000*, GMD Report 91, pages 11-28, Berlin, April 2000.

[10] B. Haverkort, H. Hermanns, and J.P. Katoen. The Use of Model Checking Techniques for Quantitative Dependability Evaluation. In *IEEE Symposium on Reliable Distributed Systems, SRDS 2000*, IEEE CS Press, October 2000.

[11] H. Hermanns, J.P. Katoen, J. Meyer-Kayser, and M. Siegle. Towards Model Checking Stochastic Process Algebra. In *IFM 2000*, Springer LNCS 1945: 420–439, November 2000.

[12] H. Hermanns, J.P. Katoen, J. Meyer-Kayser, and M. Siegle. Implementing a Model Checker for Performability Behaviour. In *Fifth International Workshop on Performability Modeling of Computer and Communication Systems*, Erlangen, September 2001. To appear.

[13] H. Hermanns, U. Herzog, U. Klehmet, V. Mertsiotakis and M. Siegle. Compositional performance modelling with the TIPPtool. *Performance Evaluation*, **39**(1-4): 5–35, 2000.

[14] J.-P. Katoen, M. Kwiatkowska, G. Norman, and D. Parker. Faster and symbolic CTMC model checking. In *PAPM/PROBMIV'01*, LNCS. Springer, 2001. To appear.

[15] W. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton Univ. Press, 1994.