Maybe or Maybe not

Contributions to Stochastic Verification

Ralf Wimmer
University of Freiburg

Erika Ábrahám
RWTH Aachen University

We summarize the contributions of the Chair for Computer Architecture to the field of verification for stochastic systems. The contributions are twofold: on the one hand, there are methods for symbolic state space minimization, on the other hand efficient algorithms for counterexample generation.

1 Introduction

The *verification* of embedded systems is crucial, especially in safety-critical application areas like avionics and automotive control, or process control in the chemical industry. Errors in the design of such embedded systems can have fatal consequences. However, the growing number of communicating modules and the increasing complexity of the tasks to be solved make the application of formal methods highly challenging. In contrast to standard microprocessors for desktop

computers, embedded systems often control continuous quantities with the help of a digital controller, resulting in so-called hybrid systems. They communicate over unreliable media like radio links and use randomized communication protocols. If used in inaccessible places like satellites, the finite lifetime of components has to be taken into account when designing an embedded system. Buffers have to be dimensioned such that they have enough capacity to store the incoming requests in at least, say, 99.9 % of the cases. Common to these examples is the need to take *stochastic* aspects into consideration during verification. To do so, these systems can be modeled as *Markov chains* or *Markov decision processes* with either discrete or continuous time.

The transregional research center AVACS, established in 2005 at the universities of Freiburg, Saarbrücken, and Oldenburg as well as the Max-Planck-Institute for Computer Science in Saarbrücken, is devoted to the development of verification methods for such complex systems. Considered aspects range from hybrid via real-time to stochastic systems. Sub-project S3 in particular considers stochastic models, which we focus on in this article. Bernd Becker's Chair of Computer Architecture contributes to this subproject with extensive knowledge and experience in the fields of symbolic methods like BDD- or SAT-based algorithms for verification of digital circuits. Due to the success of BMC-based methods in this context, a DFG project called CEBug was proposed and and established in co-operation with the RWTH Aachen University. Its goal was to investigate the generation of counterexamples for Markov models.

The main contributions can be divided into two groups: the first one considers *symbolic minimization* of Markov models. The goal is to find, for a given Markov chain or Markov deci-

sion process, a minimal[1] abstract model that shows the same observable behavior as the original model. Minimality thereby depends on what is considered observable if the model can carry out internal actions: Are they observable like external actions? Can one count the number of subsequent internal steps or only see that internal steps are executed? Such different notions of observability lead to different minimal systems.

The second contribution encompasses methods for computing *counterexamples for Markov models* in the case that some desired properties are violated. Such counterexamples are useful for reproducing errors during debugging and for abstraction refinement when an abstraction is too coarse to prove (or refute) a given property. While model checking algorithms for linear-time properties of digital systems can return a trace causing a violation of the property with no additional overhead, algorithms for the analysis of stochastic systems are not able to do the same. Let us assume that the property "The probability to eventually reach a safety-critical state is at most 0.01" is violated, i. e., the probability is actually larger than 0.01. To certify this violation, the existence of traces has to be shown whose joint probability exceeds the bound. In general, a single trace is not sufficient for violation, but a potentially large (or even infinite) set of traces is needed. This bears two challenges: (1) how to handle large state spaces and (2) how to handle large sets of paths.

In the following Section 2 we give two examples for discrete-time Markov models. The following two Sections 3 and 4 present the contributions on state space minimization and counterexamples, respectively, in more detail. Finally we

[1] minimal in the number of states

conclude this overview paper in Section 5.

2 Modeling Stochastic Systems

Markov models are a widely used formalism to model randomized systems. According to the properties of interest, either discrete- and continuous-time models can be used. Furthermore, there are models with pure probabilistic/stochastic behavior and models supporting also non-determinism.

Discrete-time Markov chains (*DTMCs*) are purely probabilistic models. They extend Kripke structures by labeling each transition with a probability value such that the probabilities on all outgoing transitions of a state sum up to 1. Time is modeled discretely by defining each transition to take one time unit.

An example of a DTMC is given in Figure 1. It models the behavior of Bernd Becker at work in the morning. Initially, he is in his office. He can leave his office only through the secretariat, therefore this transition is labeled with probability 1. Being in the secretariat, with probability 0.1 he has to leave for a meeting, and with probability 0.1 he has forgotten something to arrange or the phone is ringing and he goes back to his office. With the remaining probability 0.8 he goes towards the coffee room and on the way he visits the rooms of his chair. Going from one room to the next he might think about some research problems just discussed in the previous room. Having arrived at the next room, sometimes he already has the answer and he goes back to discuss it with probability 0.1, otherwise he proceeds to the next room. An exception is the room of Peter, responsible for the technical support, who sometimes has nice new devices, therefore the probability to

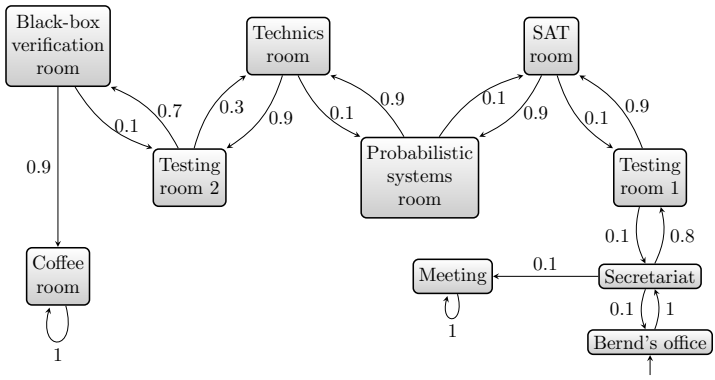


Figure 1: A DTMC model for the morning coffee of Bernd Becker at work

go back to him is higher.

In this model, the probability of finally having a morning coffee is approximately 0.88.

Markov decision processes (*MDPs*) extend DTMCs to model non-deterministic behavior. For an illustration consider the example MDP in Figure 2, which provides a model for the choice of Bernd Becker where to have lunch during work. Non-determinism is used to model actions before lunch, determining from where he starts for lunch, what in turn effects the probabilitistic choice where to go for lunch. In state "Working in the office" first a non-deterministic choice takes place: he either directly goes for lunch, or he goes to a meeting in the city center, or he goes to give a lecture. If he goes to have lunch directly from his office, he goes to ISE

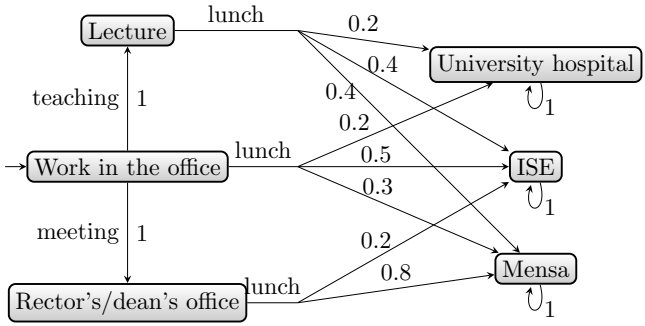


Figure 2: An MDP model for the lunch behavior of Bernd Becker

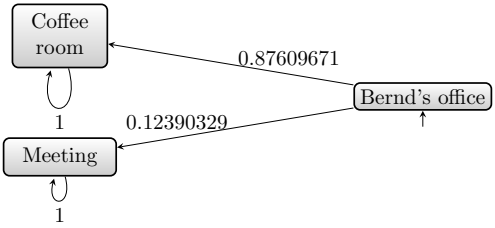
(Institute of Solar Energy Systems) with probability 0.5, to the university mensa with probability 0.3 and to the mensa at the university hospital with probability 0.2. The latter probability is lower than the other ones because the way is longer. The case for having a lecture before lunch is similar. However, if he has a previous meeting at the city center, the university hospital is too far away, therefore he goes either to ISE or to the mensa.

3 State Space Minimization

A central problem which limits the applicability of verification methods to realistic systems is the size of their state space: It grows exponentially in the number of parallel system components, making it impossible to explicitly store and manipulate

the state space. A popular technique to fight this state space explosion problem is *bisimulation-based minimization*. Its goal is to compute a (state-)minimal system whose observable behavior coincides with the observable behavior of the original system. Depending on how observability is defined, different minimal systems can be obtained. Checking properties of the original system can then be carried out on the minimal system instead.

Example 1 *Assume again the DTMC in Figure 1 and assume that only the reachability of the states* Coffee room *and* Meeting *is observable. Then the following DTMC is observably equivalent to the one in Figure 1:*



Minimizing the system first, however, is only fruitful if the minimization algorithm scales to very large systems, which are out of scope of the model checking algorithms. For this purpose, Bernd Becker et al. developed novel *symbolic* minimization methods.

The foundation was an algorithm by Blom and Orzan for minimizing labeled transition systems w. r. t. strong and branching bisimulation. This algorithm uses explicit data

structures (essentially lists of states and transitions), but allows parallelization. This algorithm is well suited for the application of symbolic methods—in this case based on binary decision diagrams (BDDs). The algorithm together with several improvements was described in a series of publications:

- [1] presents a generic symbolic framework for the minimization of labeled transition system w. r. t. all relevant bisimulations known from the literature. Optimizations of this algorithm, which considerably reduce its running time in practice, are given in [2].
- The minimization algorithm was generalized to interactive Markov chains in [3]. Numerical issues of stochastic bisimulation computation were analyzed and solved in [4].
- [5] finally presents a variant of the algorithm which automatically adapts itself to the available memory by dynamically switching between two different representations of the current state space partition.

The techniques developed for minimization were used in two applications:

- The analysis of industrial designs regarding their reliability is studied in [6]. The design, given as a STATECHART, is extended with stochastic timing information, then transformed from a labeled transitions system via an interactive Markov chain into a uniform continuous-time Markov decision process. The latter model allows to compute, e. g., time-bounded reachability probabilities. The major problem of this approach was the state space explosion: Typical benchmark models consisted of up

to 10^{20} states, which is far out of reach for numerical probability computation. Among other techniques, symbolic minimization helped reduce the relevant state space to about 10^6 states, which can be analysed with state-of-the-art algorithms.

- The goal of [7] was to compute maximal (or minimal) long-run average rewards for large MDPs. The standard algorithm to compute these values is a kind of policy iteration on the MDP. The numerical analysis is thereby performed on the DTMC induced by the current scheduler and the most expensive step. While updating the current policy and computing the induced DTMC can be done symbolically in an efficient way, symbolic methods for numerical analysis are not as successfull. Therefore we decided to perform a symbolic minimization step before such that the resulting DTMC is small enough to be analysed using explicit methods.

Applications: [7]

4 Counterexamples

Model checking for probabilistic (PCTL or more general ω-regular) properties of Markov models can be led back to the computation of reachability probabilities. Assume for example a DTMC and a property that the probability of reaching a given target state in the DTMC is at most a given upper bound. If this property is violated by the DTMC then a *counterexample* is a (possibly infinite) set of finite paths of the DTMC, all starting in the initial state and reaching the

target state, such that their total probability mass is larger than the given upper bound.

Example 2 *Consider again the DTMC in Figure 1 and the property that the probability that Bernd Becker never reaches the coffee room but ends up in a meeting instead is at most* 0.1. *Unfortunately, this property is violated by the model. The paths*

$$\text{Bernd's office} \xrightarrow{1} \text{Secretariat} \xrightarrow{0.1} \text{Meeting } \textit{and}$$
$$\text{Bernd's office} \xrightarrow{1} \text{Secretariat} \xrightarrow{0.1} \text{Bernd's office} \xrightarrow{1} \text{Secretariat} \xrightarrow{0.1} \text{Meeting}$$

together build a counterexample.

Bernd Becker et al. developed different algorithms and techniques to compute counterexamples for Markov models:

- The computation of counterexamples in a user-friendly *hierarchically-abstracted representation* is the topic of the publications [8, 9]. Instead of a set of paths, we determine a *sub-system* (a part of the model) of an abstraction such that the paths running inside of this sub-system already violate the property, i.e., they build a counterexample. Abstraction refinement can be used to stepwise concretize the counterexample at relevant places. The methods are implemented in the COMICS tool [10].

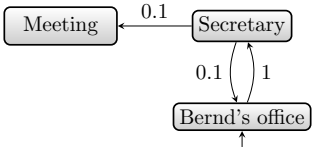
- The above approaches apply k shortest paths search in the computations. Instead, *bounded model checking* and SMT solving is used for the path search in [11, 12].

- The works [13, 14] aim at the computation of *minimal* sub-systems as counterexamples using MILP (mixed

integer linear programming) solving. A general framework allows to define different metrics for minimality, for example based on the number of states in the low-level Markov model or on the number of probabilistic guarded commands (as supported by the popular PRISM model checker to define high-level system models) which are necessary to define an erroneous sub-system and thus a counterexample.

- Finally, counterexample generation for large systems by using *symbolic* computations is discussed in [15].

Example 3 *The counterexample from Example 2 can be represented by the sub-system consisting of the states appearing in the given paths and the transitions between them:*



Note that this sub-system builds the closure of the path set regarding the involved states and transitions. E.g., all paths of the form

$$(\text{Bernd's office} \xrightarrow{1} \text{Secretary} \xrightarrow{0.1})^* \text{Meeting}$$

are contained in it.

5 Conclusion and Future Work

We gave a short high-level overview on the scientific contributions of Bernd Becker in the area of formal methods for probabilistic systems. These contributions are the result of long-term fruitful co-operations, which we always enjoyed and hope to have the possibility to continue in the future.

References

[1] Wimmer, R., Herbstritt, M., Hermanns, H., Strampp, K., Becker, B.: Sigref – A symbolic bisimulation tool box. In: Proc. of ATVA'06. Vol. 4218 of LNCS, Springer-Verlag (2006) 477–492

[2] Wimmer, R., Herbstritt, M., Becker, B.: Optimization techniques for BDD-based bisimulation minimization. In: Proc. of GLSVLSI'07, ACM Press (2007) 405–410

[3] Wimmer, R., Hermanns, H., Herbstritt, M., Becker, B.: Towards symbolic stochastic aggregation. Reports of SFB/TR 14 AVACS 16 (2007) ISSN: 1860-9821, online available at `http://www.avacs.org`.

[4] Wimmer, R., Becker, B.: Correctness issues of symbolic bisimulation computation for Markov chains. In: Proc. of MMB & DFT'10. Vol. 5987 of LNCS, Springer-Verlag (2010) 287–301

[5] Wimmer, R., Derisavi, S., Hermanns, H.: Symbolic partition refinement with automatic balancing of time and space. Performance Evaluation **67**(9) (2010) 815–835

[6] Böde, E., Herbstritt, M., Hermanns, H., Johr, S., Peikenkamp, T., Pulungan, R., Rakow, J., Wimmer, R., Becker, B.: Compositional dependability evaluation for STATEMATE. IEEE Transactions on Software Engineering **35**(2) (2009) 274–292

[7] Wimmer, R., Braitling, B., Becker, B., Hahn, E.M., Crouzen, P., Hermanns, H., Dhama, A., Theel, O.: Symblicit calculation of long-run averages for concurrent probabilistic systems. In: Proc. of QEST'10, IEEE Computer Society Press (2010) 27–36

[8] Ábrahám, E., Jansen, N., Wimmer, R., Katoen, J.P., Becker, B.: DTMC model checking by SCC reduction. In: Proc. of QEST'10, IEEE Computer Society Press (2010) 37–46

[9] Jansen, N., Ábrahám, E., Katelaan, J., Wimmer, R., Katoen, J.P., Becker, B.: Hierarchical counterexamples for discrete-time Markov chains. In: Proc. of ATVA'11. Vol. 6996 of LNCS, Springer-Verlag (2011) 443–452

[10] Jansen, N., Ábrahám, E., Volk, M., Wimmer, R., Katoen, J.P., Becker, B.: The COMICS tool – Computing minimal counterexamples for DTMCs. In: Proc. of ATVA'12. Vol. 7561 of LNCS, Springer-Verlag (2012) 349–353

[11] Wimmer, R., Braitling, B., Becker, B.: Counterexample generation for discrete-time Markov chains using bounded model checking. In: Proc. of VMCAI'09. Number 5403 in LNCS, Springer-Verlag (2009) 366–380

[12] Braitling, B., Wimmer, R., Becker, B., Jansen, N., Ábrahám, E.: Counterexample generation for Markov chains using SMT-based bounded model checking. In: Proc. of FMOODS/FORTE'11. Vol. 6722 of LNCS, Springer-Verlag (2011) 75–89

[13] Wimmer, R., Becker, B., Jansen, N., Ábrahám, E., Katoen, J.P.: Minimal critical subsystems for discrete-time Markov models. In: Proc. of TACAS'12. Vol. 7214 of LNCS, Springer-Verlag (2012) 299–314

[14] Wimmer, R., Jansen, N., Vorpahl, A., Ábrahám, E., Katoen, J.P., Becker, B.: High-level counterexamples for probabilistic automata. In: Proc. of QEST'13. Vol. 8054 of LNCS, Springer-Verlag (2013) 18–33

[15] Jansen, N., Ábrahám, E., Zajzon, B., Wimmer, R., Schuster, J., Katoen, J.P., Becker, B.: Symbolic counterexample generation for discrete-time Markov chains. In: Proc. of FACS'12. Vol. 7684 of LNCS, Springer-Verlag (2012) 134–151