CASPA: Symbolic Model Checking of Stochastic Systems

Matthias Kuntz, Markus Siegle

Universität der Bundeswehr München
Institut für Technische Informatik

Abstract. This note describes the new features of the performance evaluation tool CASPA, which has been extended by algorithms for the model checking of stochastic systems. CASPA uses stochastic process algebras as its input language. Multi-terminal binary decision diagrams (MTBDD) are employed to represent the transition system underlying a given process algebraic specification. The specification of performability requirements can be done using a newly developed stochastic temporal logic. All phases of modelling, from model construction to model checking and numerical analysis, are based entirely on MTBDDs.

1 Introduction

Symbolic data structures, such as binary decision diagrams (BDD) and variants thereof have proved to be suitable for the efficient generation and compact representation of very large state spaces and transition systems. It has been shown that in the context of a compositional model specification formalism such as process algebra, the size of the symbolic representation can be kept within linear bounds, even if the underlying state space grows exponentially [3]. The key to such compact representation is the exploitation of the compositional structure of a given specification. It is also known that in addition to functional analysis, performance analysis and the verification of performability properties can also be carried out on such symbolic representations [5].

In this note, we describe the symbolic stochastic model checking tool CASPA [4] which offers a Markovian stochastic process algebra language for model specification. CASPA generates a symbolic model representation, which is based on multi-terminal binary decision diagrams (MTBDD) [2], directly from the high-level model, without generating transition systems as an intermediate representation. In addition to specifying the model, CASPA allows the user to specify complex requirements that a system has to satisfy by means of the stochastic temporal logic SPDL (stochastic propositional dynamic logic) [4]. Model checking and numerical analysis are also carried out directly on the symbolic representation of the transition rate matrix of the underlying labelled CTMC. To our knowledge, CASPA is the first stochastic process algebra and model checking tool whose implementation relies completely on symbolic data structures.

```
/*** rate and constant definitions ***/
rate lambda=0.5;  rate gamma=0.1;  rate omega=0.4;  rate delta=1.5;  rate mu=4.3;
int max=5000;

/*** system specification ***/
System       := Arr(0) |[faulty_arr,corr,ncorr]| Errorhandler
Arr(i [max]) := [i < max] -> (arr,lambda);Arrival(i+1) + (faulty_arr,mu);Fault_Arr(i)
                [i = max] -> (prc,omega);Arrival(0)
Fault_Arr(i [max]) := (corr,1);Arr(i+1) + (ncorr,1);(rt,kappa);Arr(i)
Errorhandler := (faulty_arr,1);((corr,gamma);Errorhandler + (ncorr,delta);Errorhandler)

/*** requirement specification   (note that "!" denotes logical negation) ***/
spdl req1 P(> 0.9)(!full [arr*;faulty_arr;ncorr;rt;arr* + arr*](0,5) full)
spdl req2 P(<= 0.85)(!full [(arr + faulty_arr;corr)*](0,10) full)
```

Fig. 1. Specification of a fault-tolerant packet collector

2 The Modelling Language

The modelling language of CASPA is a stochastic process algebra where all actions have an exponentially distributed delay. The language provides operators for prefixing, choice, parallel composition and hiding. Infinite (i.e. cyclic) behaviour can be specified with the help of defining equations (instead of employing an explicit recursion operator). The technique used for symbolic model representation (cf. Sec. 4) works only for finite state spaces. Therefore the grammar of the input language is such that recursion over static operators (i.e. parallel composition and hiding) is not allowed, which ensures that the underlying state space is finite. CASPA allows the specification of parameterised processes, i.e. processes which carry one ore more integer parameters. This feature is very useful for describing the behaviour of queueing, counting, or generally indexed processes. Within a parameterised process, the enabling of actions may be conditioned on the current value of the process parameters.

We demonstrate the use of the CASPA modelling language by means of a small example which is shown in Fig. 1. It is a fault-tolerant packet collector that receives a fixed number of packets and processes them together. In the first two lines the rate parameter values and the number of packets to be collected (`max`) are defined. The entire system consists of an arrival process and an error handling process. A packet arrival can be error-free (action `arr`) or erroneous (action `faulty_arr`). An error can be correctable (action `corr`) or not correctable (action `ncorr`). If it is correctable we return from the erroneous arrivals handling process to the arrival process, awaiting a new packet. If it is not correctable the respective packet has to be retransmitted (action `rt`). In the last two lines two example requirement specifications can be found. They are indicated by the key word `spdl`, and are identified by an arbitrary but unique name (`req1` resp. `req2`).

3 Specification of Requirements

Using the logic SPDL, CASPA supports the definition and verification of a very general class of performance and reliability requirements. Like the logic CSL [1], SPDL allows to reason about the transient and steady-state behaviour of stochastic systems. Also like CSL, SPDL provides both state and path formulae. Unlike

CSL, in SPDL the required system behaviour is specified in the form of action sequences , which are defined with the help of extended regular expressions (also called *programs*). These extensions of regular expressions make it possible to express programming language constructs such as $if\text{-}then\text{-}else$ and $while$. Probabilistic path-based formulae of SPDL are of the form $\mathcal{P}_{\bowtie p}(\phi_1[\rho]^t\phi_2)$, where the part inside the parentheses is interpreted as follows: Within t time units a state satisfying the state formula ϕ_2 must be reached, and all preceding states must satisfy ϕ_1. In addition, the actions on the considered path must form a word that can be induced from the program ρ. For the packet collector from Fig. 1, the given requirements read as follows:

`req1`: The probability to receive `max` data packets (with at most one non-correctable error) within 5 time units should be greater than 90%.

`req2`: The probability to reach a state in which the state formula `full` holds within 10 time units, provided no packet contains incorrectable errors, should be at most 85%.

4 Construction of the State Space Representation

CASPA translates a given process algebraic specification directly to an MTBDD-based symbolic representation of the underlying state space and transition system. It uses the CUDD library [6] which provides support for the construction and manipulation of BDD-based data structures. The translation implements the denotational semantics described in [4], with some extensions and optimisations. The basic idea of this translation is as follows: In a first step, the parse tree of the process algebraic specification at hand is constructed. Then the MTBDD representation of the underlying transition relation is constructed in a compositional fashion, starting with sequential processes (i.e. processes which do not contain the parallel composition operator) which are located close to the leaves of the parse tree. Finally the MTBDD for the overall process is built from the MTBDDs of its components by applying rules for symbolic parallel composition. This construction procedure is completely symbolic and compositional, i.e. each sub-process of the specification is represented by an MTBDD, which is then used as an operand during the construction of the higher-level processes.

5 Model Checking and Numerical Analysis

Model checking a requirement of the form $\mathcal{P}_{\bowtie p}(\phi_1[\rho]^t\phi_2)$ requires the following steps:

- For the program ρ, a non-deterministic program automaton $\mathcal{A}$ is generated.
- From $\mathcal{A}$ and the system model $\mathcal{M}$ a product Markov chain $\mathcal{M}^\times$ is generated, thereby making $\mathcal{A}$ deterministic and restricting $\mathcal{M}$ to the satisfying paths. This can be done on the purely symbolic level.
- On $\mathcal{M}^\times$ the probability with which the formula is satisfied is computed.

For the numerical computation of a system's steady-state or transient state probabilities, CASPA uses code that was developed within the PRISM project [5]. Table 1 gives the results for model checking the system from Fig. 1. The run-times were obtained on a standard PC and are given in seconds. Column "M.C. Time" includes generation of $\mathcal{M}^{\times}$ from $\mathcal{M}$ and $\mathcal{A}$ and transient numerical analysis.

`max`	States $\mathcal{M}$	`req1`		`req2`	
		States $\mathcal{M}^{\times}$	M.C. Time	States $\mathcal{M}^{\times}$	M.C. Time
5,000	20,001	10,002	2.01	5,003	1.91
15,000	60,001	30,002	7.18	15,003	8.02
30,000	90,001	60,002	15.92	30,003	15.27
50,000	150,001	100,002	29.45	50,003	29.28

Table 1. Results of model checking the system from Fig. 1

6 Conclusion and Future Work

This note briefly described the new features of the symbolic stochastic model checking tool CASPA. Based on the case studies conducted in [4] one can state that CASPA is very efficient, both with respect to state space generation and model checking. As future work, we are planning to extend the class of models which the tool can handle, to add some advanced analysis techniques and (last but not least) to improve the user interface of the tool.

References

1. C. Baier, B. Haverkort, H. Hermanns, and J.P. Katoen. Model-Checking Algorithms for Continuous-Time Markov Chains. *IEEE Trans. Software Eng.*, 29(7):1–18, 2003.
2. M. Fujita, P. McGeer, and J.C.-Y. Yang. Multi-terminal Binary Decision Diagrams: An efficient data structure for matrix representation. *Formal Methods in System Design*, 10(2/3):149–169, April/May 1997.
3. H. Hermanns, M. Kwiatkowska, G. Norman, D. Parker, and M. Siegle. On the use of MTBDDs for performability analysis and verification of stochastic systems. *Journal of Logic and Algebraic Programming*, 56(1-2):23–67, 2003.
4. M. Kuntz. *Symbolic Semantics and Verification of Stochastic Process Algebras*. PhD thesis, Universität Erlangen-Nürnberg, Institut für Informatik, 2006. (to appear).
5. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal on Software Tools for Technology Transfer (STTT)*, 6(2):128–142, 2004.
6. F. Somenzi. CUDD: Colorado University Decision Diagram Package, Release 2.3.1. User's Manual and Programmer's Manual, February 2001.