CASPA: A Tool for Symbolic Performance and Dependability Evaluation

Matthias Kuntz, Markus Siegle, University of the Federal Armed Forces Munich, Dept. of Computer Science
[kuntz,siegle]@informatik.unibw-muenchen.de
Edith Werner, Georg-August-Universität Göttingen, Dept. of Computer Science
ewerner@informatik.uni-goettingen.de

I. Introduction

Symbolic data structures, such as binary decision diagrams (BDD) [1] and variants thereof have proved to be suitable for the efficient generation and compact representation of very large state spaces and transition systems. The key to such compact representation is the exploitation of the compositional structure of a given specification [2], [3]. 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 [4], [3].

In this note, we describe the current status of our tool CASPA which offers a Markovian stochastic process algebra language for model specification. CASPA uses multi-terminal binary decision diagrams (MTBDD) [5], [6] to represent the transition systems underlying a given process algebra specification. All phases of modelling, from model construction via numerical analysis to the computation of performance measures, are based entirely on this symbolic data structure.

II. The modelling language

The modelling language of CASPA is essentially the same as the language supported by TIPPtool [7]. It is a stochastic process algebra where all actions have an exponentially distributed delay. The language provides operators for prefixing, choice, parallel composition, enabling, disabling and hiding. Infinite (i.e. cyclic) behaviour can be specified with the help of defining equations. The technique used for symbolic model representation (cf. Sec. III) 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 shown in Fig. 1. It is a queueing system, consisting of two arrival processes (inter-arrival times have Erlang-3 and Erlang-2 distribution) and a service center with finite capacity buffer and exponential service times. In the first three lines the rate parameter values and the buffer size are defined. The last three lines of the example show that CASPA supports the definition and computation of three different types of performance measures:

```
/*** rate and constant definitions ***/
rate l1 = 0.5;
rate l2 = 0.1;
rate mu = 4.3;
int max = 15;

/*** system specification ***/
System   := ( Arrival1 |[]| Arrival2 ) |[enq]| Server(0)
Arrival1 := (tau,l1); (tau,l1); (enq,l1); Arrival1
Arrival2 := (tau,l2); (enq,l2); Arrival2
Server(n [max]) := [n < max] -> (enq,1); Server(n+1)
                   [n > 0]   -> (serve,mu); Server(n-1)

/*** measure definition ***/
statemeasure Buffer_full Server(n = max)
throughputmeasure Service_rate serve
meanvalue Occupancy Server(n)
```

Fig. 1. Specification of a small queueing system example

A *state measure* is defined as the probability of the system being in a specific state or in a well-defined subset of the state space. Such sets of states are defined by referencing to process names, thereby possibly using Boolean operators and conditioning on process parameter ranges.

A *throughput measure* is defined as the mean number of occurences of a specific named action per unit of time.

A *mean value measure* is defined as the expected value of a certain process parameter, taken over all reachable states.

III. Construction of the symbolic state space representation

CASPA translates a given process algebra specification directly to an MTBDD-based symbolic representation of the underlying state space and transition system. It uses the CUDD library [8] which provides support for the construction and manipulation of BDD-based data structures. The translation implements the denotational semantics described in [9], with some extensions and optimisations. The basic idea of this translation is as follows: In a first step, the parse tree of the process algebra 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 the rules for symbolic parallel composition (see, e.g, [3]). 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.

Already during its construction, the parse tree is annotated with information concerning the performance measures to be derived. For example, nodes associated with a process name that occurs in a state measure specification are marked, and subsequently a BDD is constructed which encodes exactly the states which are relevant for that state measure. Nodes associated with a parameterised process for which a mean value has to be computed are also marked, and subsequently an MTBDD is constructed which encodes the relevant states and associated parameter value.

IV. Numerical analysis and computation of performance measures

CASPA supports steady state and transient analysis. For steady state analysis Power, Jacobi, Pseudo-Gauss-Seidel and their overrelaxed versions can be used. For transient analysis uniformisation is employed. The rate matrix of the CTMC is obtained from the symbolic representation of the transition system by abstracting from the action labels. This abstraction can easily be performed with the help of MTBDD operations, and the result is a symbolic representation of the rate matrix.

For the numerical computation of the probability vectors CASPA uses the iterative solvers code developed within the PRISM project [10], which is available from the PRISM web page. To our knowledge, these algorithms are currently the fastest available implementation of MTBDD-based linear systems solvers. They rely on the advanced concept of hybrid offset-labelled MTBDDs as described in [10].

Once the probability vector has been obtained, the performance measures of interest are computed. Basically, each type of measures can be computed as the sum or weighted sum of certain state probabilities. These expressions are computed efficiently on the basis of MTBDD operations, using the MTBDDs which we mentioned at the end of Sec. III.

As an example, we modelled the well-known Kanban system from [11] with four cells and N Kanban cards (of a single type). We considered the case that a workpiece may need rework and computed, among others, the following measures:

- Mean values: average number of cards in each cell.
- Throughput measures: throughput of parts with and without rework for each cell.

The state space size and time needed for computing the measures can be found in Table I, where MTBDD-generation times include reachability analysis and numerical analysis times include measure computation. The steady-state analysis method used here was Pseudo-Gauss-Seidel. The results were obtained on an AMD Athlon 2000+ CPU with 512 MB RAM, running Red Hat 8.0 Linux.

N	reachable states	MTBDD nodes	MTBDD generation	numerical analysis
4	454,475	3990	0.17 sec.	20.41 sec.
5	2,546,432	5392	0.42 sec.	184.44 sec.
6	11,261,376	8086	0.89 sec.	1191.46 sec.
7	41,644,800	10389	1.74 sec.	21h
8	133,865,325	13998	3.28 sec.	-
9	384,392,800	17762	5.23 sec.	-
10	1,005,927,208	23231	8.94 sec.	-

TABLE I
Results for Kanban system

V. Conclusion and future work

Based on the results of this and other case studies we can state the following: State space construction as realised in CASPA is very efficient, it is several orders of magnitude faster than that of TIPPtool which uses an explicit (and therefore much less space efficient) representation of the state space and transition system. The numerical solution with the algorithms from PRISM achieves an efficiency which is almost comparable to state-of-the-art sparse solvers. Once the state probabilities have been computed, the derivation of performance measures is very fast, in fact it is much faster than for TIPPtool.

We are currently extending CASPA to a stochastic model checker which supports specification and verification of complex performability properties of stochastic systems, specified with the help of the temporal logic sPDL [12].

References

[1] R. Bryant, "Graph-based Algorithms for Boolean Function Manipulation," *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677–691, August 1986.

[2] L. de Alfaro, M. Kwiatkowska, G. Norman, D. Parker, and R. Segala, "Symbolic Model Checking for Probabilistic Processes using MTBDDs and the Kronecker Representation," in *TACAS'2000*, S. Graf and M. Schwartzbach, Eds. Springer LNCS 1785, 2000, pp. 395–410.

[3] M. Siegle, "Advances in model representation," in *Process Algebra and Probabilistic Methods, Joint Int. Workshop PAPM-PROBMIV 2001*, L. de Alfaro and S. Gilmore, Eds. Springer, LNCS 2165, September 2001, pp. 1–22.

[4] M. Kwiatkowska, G. Norman, and D. Parker, "Probabilistic Symbolic Model Checking with PRISM: A Hybrid Approach," in *TACAS'2002*, J.-P. Katoen and P. Stevens, Eds. Springer LNCS 2280, April 2002, pp. 52–66.

[5] M. Fujita, P. McGeer, and J.-Y. Yang, "Multi-terminal Binary Decision Diagrams: An efficient data structure for matrix representation," *Formal Methods in System Design*, vol. 10, no. 2/3, pp. 149–169, April/May 1997.

[6] R. Bahar, E. Frohm, C. Gaona, G. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Algebraic Decision Diagrams and their Applications," *Formal Methods in System Design*, vol. 10, no. 2/3, pp. 171–206, April/May 1997.

[7] H. Hermanns, U. Herzog, U. Klehmet, V. Mertsiotakis, and M. Siegle, "Compositional performance modelling with the TIPPtool," *Performance Evaluation*, vol. 39, no. 1-4, pp. 5–35, January 2000.

[8] F. Somenzi, "CUDD: Colorado University Decision Diagram Package, Release 2.3.0," September 1998, user's Manual and Programmer's Manual. [Online]. Available: http://vlsi.colorado.edu/~fabio.

[9] M. Kuntz and M. Siegle, "Deriving symbolic representations from stochastic process algebras," in *Process Algebra and Probabilistic Methods, Proc. PAPM-PROBMIV'02*, H. Hermanns and R. Segala, Eds. Springer, LNCS 2399, 2002, pp. 188–206.

[10] D. Parker, "Implementation of symbolic model checking for probabilistic systems," Ph.D. dissertation, School of Computer Science, Faculty of Science, University of Birmingham, 2002.

[11] G. Ciardo and M. Tilgner, "On the use of Kronecker operators for the solution of generalized stochastic Petri nets," ICASE, Tech. Rep. 96-35, 1996.

[12] M. Kuntz and M. Siegle, "A stochastic extension of the logic PDL," in *Sixth Int. Workshop on Performability Modeling of Computer and Communication Systems (PMCCS6)*, Monticello, Illinois, 2003, pp. 58–61.