Symbolic Activity-Local State Graph Generation in the Context of Möbius

Kai Lampka, Markus Siegle [1]

Friedrich-Alexander-Universität Erlangen-Nürnberg, Institut für Informatik

Abstract

We describe a method for constructing compact representations of labelled continuous-time Markov chains which are derived from high-level model descriptions, such as stochastic Petri nets, stochastic process algebras, etc.. Our approach extends existing techniques in that symbolic (i.e. BDD-based) representations are constructed for non-modular, flat model descriptions. It is well suited for performance evaluation tools such as Möbius, where the high-level description is mapped onto the corresponding state graph in a monolithic manner. The symbolic representation of the overall model is obtained by merging symbolic activity-local representations of transitions. Such a scheme will not only yield a compact symbolic representation, it also has the advantage that the state space of the overall model may only need to be explored partially.

1 Introduction

In this note, we address the problem of generating the state graph underlying the high-level description of a performability model, specified by means of stochastic Petri nets, stochastic process algebra, etc.. The goal is to explore huge state spaces in a time-efficient manner, and store the associated labelled continuous-time Markov chain in a compact way. We use binary decision diagrams (BDD) and extensions thereof for representing the state graph symbolically. It is known that extremely compact symbolic representations can be constructed if the compositional structure of the high-level model at hand is taken into consideration [9,5,16,17]. Our approach extends existing techniques in that symbolic representations are constructed for non-modular, flat model descriptions. In the approach described here, the symbolic representation of the overall model is obtained by merging symbolic activity-local representations of transitions. Such a scheme of activity-local state graph generation

[1] {kilampka,siegle}@informatik.uni-erlangen.de

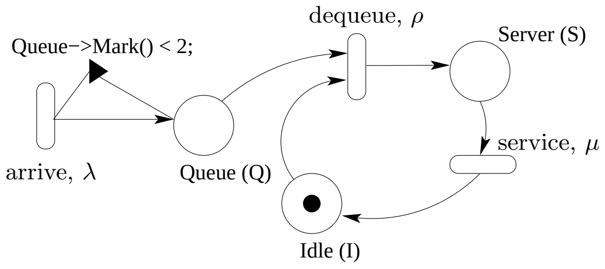

Fig. 1. A simple producer-consumer system described as a monolithic SAN

does not require the high-level description to be hierarchically structured in order to obtain a compact symbolic representation of the underlying state graph. Therefore the here illustrated approach is highly suited for performance evaluation tools such as Möbius [3], where the high-level description is mapped onto the corresponding state graph in a monolithic manner. The scheme described here will not only yield a compact symbolic representation, it also offers the advantage that the state space of the overall model may only need to be explored partially. By applying this scheme one can thus expect both runtime and memory savings.

The paper is organised as follows: In Sec. 2, we briefly introduce BDDs and MTBDDs and the general idea of symbolic state graph representation. In Sec. 3, the new scheme of activity-local state graph generation is described, and Sec. 4 concludes the paper.

2 Preliminaries

BDDs [2] are the most popular data structure for symbolic state space representation. Many extensions of BDDs have been developed in order to employ them successfully in the context of hardware verification. In the context of stochastic modelling, the most prominent BDD-based data structures are *multi-terminal BDDs* (MTBDDS), *multi-valued decision diagrams* (MDD) and *matrix diagrams* (MD) (the interested reader may find a detailed survey in [14]). In the following subsections we briefly recapitulate the basic notion of MTBDD-based state space encoding.

2.1 State spaces and their encoding

During state space exploration, a state of the overall system is represented by a state descriptor which is a vector $\vec{s}$ consisting of n elements, i.e. $\vec{s} = (s_1, \dots, s_n)$ where $s_i \in \{0, \dots, K_i\}$ for all $1 \leq i \leq n$. The vector elements are called state variables (SVs). A SV s_i normally refers to a specific place p_i in case a Petri net based modeling method is employed, or to a specific process or process parameter in case of a process algebraic model.

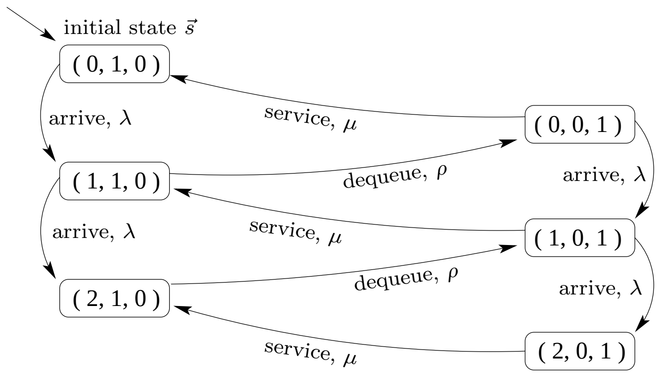

Fig. 2. State graph of the running example

The state space can be explored in a monolithic fashion by executing each enabled activity, one at a time, and determining the value of each SV s_i. As an example we consider the SAN[2] model given in Fig. 1 which will be used as a running example. Here, each state (marking of the SAN) is described by a 3-dimensional state descriptor, where each element s_i indicates the number of tokens contained in the corresponding place p_i. As shown in Fig. 1, the initial state is given by assigning $\vec{s} = (0, 1, 0)$. The whole state graph is given in Fig. 2 (the capacity of place Queue is bounded by 2).

One may encode each SV s_i by applying an injective encoding function $\mathcal{E} : \{0, \ldots, K_i\} \mapsto \mathbb{B}^{n_{s_i}}$, where we may choose $n_{s_i} \geq \lceil \log_2(K_i + 1) \rceil$. In our running example, where each state is described by a 3-dimensional vector $\vec{s}$, the encoding of the initial state is given by the Boolean vector $\vec{b} = (00, 1, 0)$. One may also encode the activity labels in a similar way: If there are K_{Act} different activities, we can define a function $\mathcal{I} : \mathcal{A}ct \mapsto \{0, \ldots, K_{Act} - 1\}$ which returns an index for each activity. We can then encode the index of each activity by applying an encoding function $\mathcal{E}$ on the set of activity indices.

For encoding the state graph, Boolean vector $\vec{b}$ encodes the values of the SVs before (source state) and $\vec{b}\,'$ (target state) after an activity's execution, and the activity label is encoded by the Boolean vector $\vec{a}$. The execution of an activity l is thus encoded by the following scheme:

$$\Big((s_1, \ldots, s_n) \xrightarrow{l} (s'_1, \ldots, s'_n)\Big) \equiv (\vec{a}, \vec{b}, \vec{b}\,')$$

Note that, for simplicity, we concentrate on tangible states only, i.e. we assume that vanishing states are eliminated "on-the-fly". The interested reader may refer to [10], the approach described there will work in our context as well.

[2] Stochastic Activity Network [15]

2.2 MTBDD-based state graph representation

MTBDDs [8,1] are an extension of BDDs for the graph-based representation of pseudo-Boolean functions. I.e. an MTBDD M is a canonical representation of a function of type $f_{\mathsf{M}} : \mathbb{B}^n \mapsto \mathbb{D}$, where $\mathbb{D}$ is a finite set. An MTBDD is a collapsed binary decision tree whose isomorphic subtrees have been merged and whose don't-care vertices are skipped. We consider ordered MTBDDs where on every path from the root to a terminal vertex the variable labeling of the nonterminal vertices obeys a fixed ordering. In the sequel we assume that the MTBDD variables have the following ordering, denoted by $\prec$: At the first n_A levels from the root are the variables a_i, encoding the activity labels. On the remaining levels we have $2n_s := 2\sum_{i=1}^{n} n_{s_i}$ variables, encoding the source and target values of the n SVs. In order to obtain small MTBDD sizes, the variables b_i and b'_i are ordered in an interleaved fashion, yielding the following overall variable ordering [3]:

$$\mathsf{a}_1 \prec \ldots \mathsf{a}_{n_A} \prec \mathsf{b}_1 \prec \mathsf{b}'_1 \prec \ldots \prec \mathsf{b}_{n_s} \prec \mathsf{b}'_{n_s}$$

Given two MTBDDs M_1 and M_2 and an arithmetic operator $\star \in \{+, -, *, \ldots\}$, we simply write $\mathsf{M} := \mathsf{M}_1 \star \mathsf{M}_2$ to obtain the MTBDD which represents $f_{\mathsf{M}_1} \star f_{\mathsf{M}_2}$. These standard arithmetic (and Boolean) operators can be implemented efficiently on the MTBDD data structure with the help of the so-called APPLY algorithm [8].
The table in Fig. 3 (A) shows the binary encoding of the state graph of Fig. 2, and part (B) shows its symbolic representation by means of an MTBDD. In the MTBDD, a dashed (solid) line indicates the value 0 (1) of the corresponding Boolean variable.

3 Symbolic activity-local state graph generation

The main idea of the approach presented here is the conventional generation of the state graph, where each transition is encoded symbolically and inserted into an MTBDD associated with the current activity. Afterwards, the symbolic representation of the overall state graph can be obtained by merging the sets of activity-local transitions.

3.1 Partitioning of the state descriptor

In the style of Petri net based modeling methods, we can now record that the execution of an activity l depends on a set of SVs, denoted as *pre*-set ($\bullet S_l$), and that the execution itself will change a set of SVs, denoted as *post*-set ($S_l\bullet$). The union of these sets yields the set of dependent SVs, $S_{d_l} := \bullet S_l \cup S_l\bullet$, and

[3] This interleaved ordering is the commonly accepted heuristics for obtaining small MTBDD sizes, see for instance [7,8,16].

(A) Binary encoding of the state graph of Fig. 2

$\vec{a}$	$\mathsf{b}_1\mathsf{b}_2$	b_3	b_4	$\mathsf{b}'_1\mathsf{b}'_2$	b'_3	b'_4	f_M
00	00	1	0	01	1	0	λ
00	01	1	0	10	1	0	λ
00	00	0	1	01	0	1	λ
00	01	0	1	10	0	1	λ
01	01	1	0	00	0	1	ρ
01	10	1	0	01	0	1	ρ
10	00	0	1	00	1	0	μ
10	01	0	1	01	1	0	μ
10	10	0	1	10	1	0	μ

(B) Corresponding symbolic representation by MTBDD M

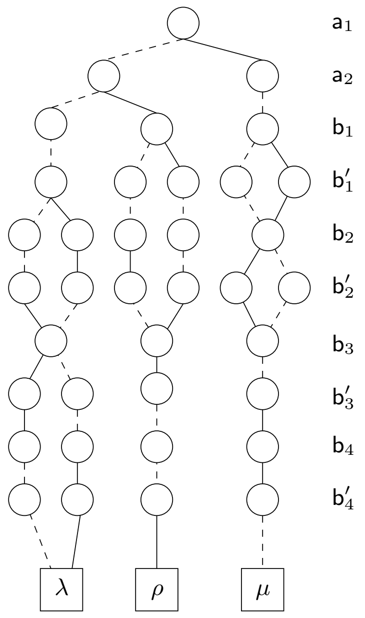

Fig. 3. Binary encodings of an SLTS and corresponding MTBDD

its complement is denoted as S_{c_l}. The elements of S_{c_l} are those SVs which are neither affected by l's execution, nor do they influence its enabling condition.

This concept of dependent and independent SVs yields the following encoding scheme for a transition induced by an activity l:

$$\Big((S_{d_l}, S_{c_l}) \xrightarrow{l} (S'_{d_l}, S_{c_l})\Big) \equiv (l, S_{d_l}, S'_{d_l}),$$

where S_{d_l} refers to the SV before and S'_{d_l} after l's execution. S_{c_l} can be omitted, since its elements are immaterial for l's execution. The possible values of S_{c_l} and S'_{c_l} will be inserted later during the stage of symbolic completion and composition.

We can apply the concept of pre- and post-set directly to the Boolean vectors which encode the state variables. For our running example, the activity-dependent binary encodings as well as the activity-dependent sets of Boolean vectors S_{d_l} and S_{c_l} are given in Table 1. For example, the execution of activity *arrive* yields the following encoding scheme:

$$\Big((Q, I, S) \xrightarrow{arrive} (Q', I, S)\Big) \equiv (00, \mathsf{b}_1\mathsf{b}_2, \mathsf{b}'_1\mathsf{b}'_2)$$

3.2 Generation of the symbolic state graph representation

The idea behind activity-local state graph generation is as follows: At the first stage, K_{Act} MTBDDs M_l are constructed. These encode the set of dependent

activity	j	$\vec{\mathsf{a}}$	$S_{d_l} := \bullet S_l \cup S_l \bullet$	$S_{c_l} := S \setminus S_{d_l}$
arrive	0	00	$\mathsf{b}_1, \mathsf{b}_2$	$\mathsf{b}_3, \mathsf{b}_4$
dequeue	1	01	$\mathsf{b}_1, \mathsf{b}_2, \mathsf{b}_3, \mathsf{b}_4$	$\emptyset$
service	2	10	$\mathsf{b}_3, \mathsf{b}_4$	$\mathsf{b}_1, \mathsf{b}_2$

Table 1
Activities, their encoding and their sets S_{d_l} and S_{c_l}

SVs (S_{d_l}) before and after a specific execution of activity l. Once all transitions are generated, each of these activity-local MTBDDs M_l needs to be supplemented (completed) by its set of symbolically encoded not-dependent SVs S_{c_l}, yielding the symbolic representation of the set of potential transitions induced by activity l. Finally the K_{Act} supplemented MTBDDs, denoted $\widetilde{\mathsf{M}_l}$, must be merged in order to encode the transition relation of the overall model. A symbolic reachability analysis needs to be carried out then, in order to restrict the potential transition relation to the actually reachable states.

3.2.1 Layout of the state graph generation procedure

For gaining as much flexibility as possible, we propose to break the major task of conventional state space exploration and symbolic encoding into two parts:

(i) A conventional state space exploration algorithm finds all transitions between reachable states, by successively firing all enabled transitions, one at a time, for each detected state descriptor. As a consequence the algorithm needs to operate on two data structures:
  (a) A state buffer, containing the already detected but yet not explored states.
  (b) A transition buffer, holding detected transitions of the form $(\vec{s}, l, \lambda, \vec{s}')$ which are to be entered into the activity-local MTBDD M_l, where $\vec{s}$ and $\vec{s}'$ are the state descriptors before and after the execution of activity l, and where λ is the rate of the activity.

(ii) This conventional exploration part is complemented by an administration part. This administration module collects and encodes the detected transitions from the transition buffer and inserts them into the activity-local MTBDDs. Furthermore, it must decide whether a state needs to be entered into the state buffer or not, i.e. the symbolic encoding module may prevent the exploration part from doing redundant work. Since the state space should be only visited partially, in order to save time, the conditions of inserting a state into the state buffer need to be considered carefully.

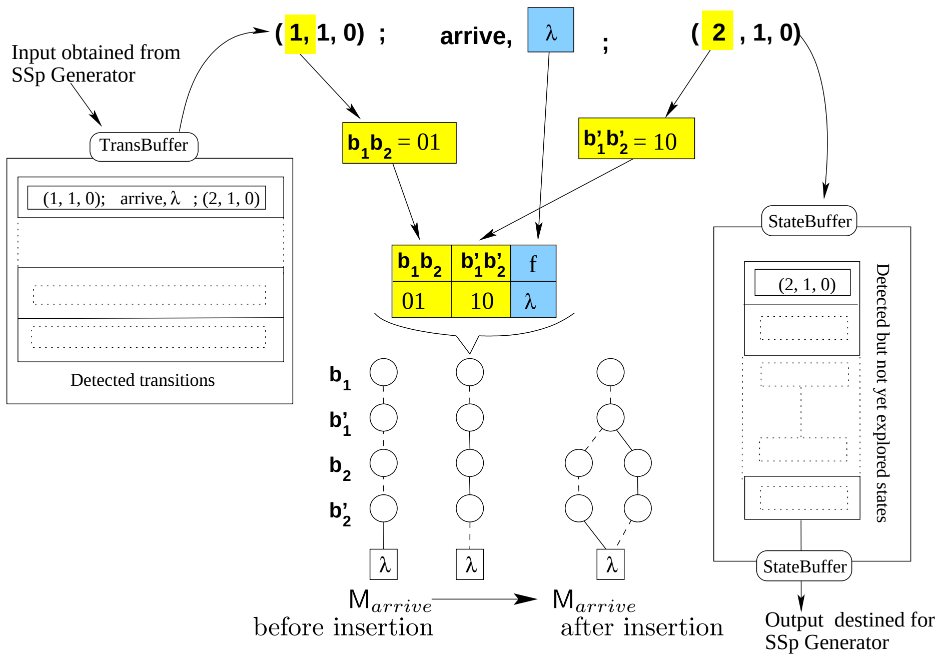

Fig. 4. Symbolic encoding of transitions

The individual tasks of generating and merging symbolic state graph representations will be discussed in greater detail now.

3.2.2 Generating the symbolic activity-local state graphs

The procedure of generating activity-local MTBDDs takes a transition $(\vec{s}, l, \lambda, \vec{s}')$ from the transition buffer. It encodes the dependent SVs S_{d_l} in binary form and inserts the respective path into the MTBDD M_l, where the rate λ is stored in a terminal node. In terms of our running example the procedure is illustrated in Fig. 4, where $l = arrive$, $\vec{s} := (1, 1, 0)$ and $\vec{s}' := (2, 1, 0)$. We propose to employ temporarily unreduced MTBDDs at this stage, in order to minimize the runtime of the insertion procedure [4]. This makes sense, since at this stage of state space generation the waste of memory space, due to redundant substructures within the MTBDDs, is acceptable. The state space explosion will manifest itself only in the later step of constructing a model's potential transition relation, where the MTBDDs will be reduced by applying the common reduction rules during completion and merging of the activity-local MTBDDs. The algorithm for inserting the MTBDD-based representation of an activity-local transition into M_l can be sketched as follows:

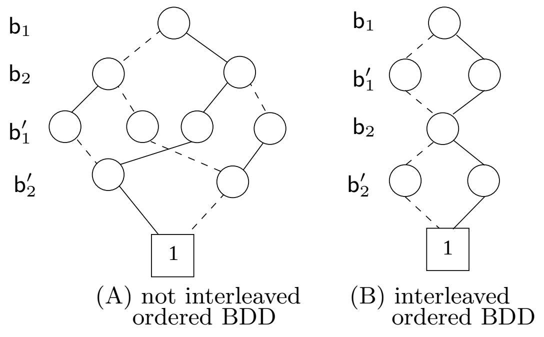

Fig. 5. Representation of BDD $Stab_{service}$

```
(1) SymbolicEnc($\mathsf{M}_l, \vec{s}_{d_l}, l, \lambda, \vec{s}_{d_l}\,'$)
(2)     $NewTrans := \mathcal{M}(\vec{b}; \mathcal{E}(\vec{s}_{d_l})) \wedge \mathcal{M}(\vec{b}\,'; \mathcal{E}(\vec{s}_{d_l}\,'))$
(3)     $NewTrans := NewTrans \cdot \lambda$
(4)     $\mathsf{M}_l := \mathsf{M}_l + NewTrans$
(5)     return $\mathsf{M}_l$
```

Hereby $\mathsf{M}_l := 0$ before the first insertion. The vectors $\vec{s}_{d_l}$ and $\vec{s}_{d_l}\,'$ encode the values of the dependent SVs before and after the execution of l. Parameter λ is the rate between these two states, which will be stored in a terminal vertex of M_l. $\mathcal{M}$ is the minterm function which constructs the conjunction of n literals given as first argument (e.g. $\vec{b}$) according to the value given as second argument (e.g. $\mathcal{E}(\vec{s}_{d_l})$).

3.2.3 Merging of activity-local MTBDDs

After the activity-local MTBDDs have been generated, the overall state graph can be generated by merging the activity-local MTBDDs. Before the actual merging, the symbolic encoding of the set S_{c_l} (complements of the set of dependent SVs) must be inserted into each MTBDD M_l. For the variables from S_{c_l} the condition $\vec{b} = \vec{b}'$ must hold, because under activity l the SVs $s_i \in S_{c_l}$ do not change their value, they stay stable. This condition is achieved by the BDD $Stab_l$ defined as follows:

$$f_{Stab_l}(\vec{b}, \vec{b}\,') := \bigwedge_{\forall \mathsf{b}_i \in S_{c_l}} (\mathsf{b}_i = \mathsf{b}'_i)$$

It is interesting to note that the interleaved ordering of the variables minimises the number of vertices of $Stab_l$ [2,7,16]. For our running example, the BDD $Stab_{service}$ (which encodes stability of SV Queue) is illustrated in Fig. 5.

In order to enable the calculation of impulse rewards, we insert the encoding of the activity labels (represented by BDD $\mathcal{M}(\vec{\mathsf{a}}, \mathcal{E}(\mathcal{I}(l)))$ into M_l. Then we can sum the K_{Act} activity-local MTBDDs. The whole process of completion and merging is thus given by:

$$\mathsf{M} := \sum_{l \in \mathcal{A}ct} \mathcal{M}(\vec{\mathsf{a}}, \mathcal{E}(\mathcal{I}(l))) \cdot \widetilde{\mathsf{M}}_l = \sum_{l \in \mathcal{A}ct} \mathcal{M}(\vec{\mathsf{a}}, \mathcal{E}(\mathcal{I}(l))) \cdot \mathsf{M}_l \cdot \mathit{Stab}_l,$$

One may note that the entire scheme of activity-local state graph generation can be interpreted as the partitioning of the overall model into K_{Act} submodels, where each submodel consists of one activity and the set of its dependent SVs S_{d_l}. The process of merging the symbolically encoded not necessarily connected activity-local state graphs can then be understood as the execution of a Möbius-type Join of the K_{Act} submodels [12].

The MTBDD M thus constructed encodes the potential set of transitions of the overall model. One needs to carry out a reachability analysis, in order to eliminate transitions originating from non-reachable states. A symbolic reachability analysis can be performed as follows [17]:

```
(1)  SymbolicReachabilityAnalysis($\mathsf{M}(\vec{a}, \vec{b}, \vec{b}\,')$)
(2)    $Trans(\vec{b}, \vec{b}\,') := Abstract(\mathsf{M}, \vec{\mathsf{a}}, +)$
(3)    $Reach(\vec{b}\,') := \mathcal{M}(\vec{b}\,', \mathcal{E}(s_1))$
(4)    $Unex(\vec{b}\,) := \mathcal{M}(\vec{b}, \mathcal{E}(s_1))$
(5)    while $Unex(\vec{b}\,) \neq 0$ do
(6)        $New(\vec{b}\,') :=$ Abstract$((Trans(\vec{b}, \vec{b}\,') \wedge Unex(\vec{b}\,)), \vec{b}, \vee) \wedge \overline{Reach(\vec{b}\,')}$
(7)        $Reach(\vec{b}\,') := Reach(\vec{b}\,') \vee New(\vec{b}\,')$
(8)        $Unex(\vec{b}\,) := New(\vec{b}\,')\{\vec{b}\,' \leftarrow \vec{b}\}$;
(9)    od
(10)   return $Reach(\vec{b}\,')$
```

One should note that reduction of the state space to the set of reachable states may increase the size of the corresponding MTBDD, which is due to the loss of regularity [16].

4 Summary and Future work

In this paper we have shown how symbolic state graph representations can be constructed in the context of monolithic models such as Möbius. We saw that only parts of the state graph need to be generated, which may yield advantages concerning run-time behaviour. The complete state graph of the overall model is constructed by merging the activity-local state graphs and subsequent symbolic state space exploration. Furthermore, employing unreduced MTBDDs during the encoding phase of the transitions may reduce the runtime considerably.

So far, we have not considered the problem that the bounds K_i for the state variables may not be known a priori. However, we have considered to employ

Z-BDDs [13] and their extension to the multi-terminal case. The reduction rules for this type of decision diagram enables an efficient handling of this problem, and it also reduces the memory requirements for representing f_{Stab} [11].

The performance evaluation tool Möbius is capable of exploiting symmetries specified within a model, generating a reduced state space by applying the lumpability theorem on-the-fly. We plan to support this feature during the symbolic state space generation as well. Furthermore, we also plan to develop an efficient scheme for handling reward variables in the symbolic context, especially in combination with reduced overall models.

References

[1] R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic Decision Diagrams and their Applications. *Formal Methods in System Design*, 10(2/3):171–206, April/May 1997.

[2] R.E. Bryant. Graph-based Algorithms for Boolean Function Manipulation. *IEEE ToC*, C-35(8):677–691, August 1986.

[3] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. Webster. The Möbius Modeling Tool. In de Alfaro and Gilmore [6], pages 241–250. Proc. of the Joint International Workshop, PAPM-PROBMIV 2001, Aachen, Germany.

[4] I. Davies, W.J. Knottenbelt, and P.S. Kritzinger. Symbolic Methods for the State Space Exploration of GSPN Models. In *Proc. of the 12th Int. Conf. on Modelling Techniques and Tools (TOOLS 2002)*, LNCS 2324, pages 188 – 199. Springer, April 2002.

[5] 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 S. Graf and M. Schwartzbach, editors, *TACAS'2000, LNCS 1785*, pages 395–410, Berlin, 2000.

[6] Luca de Alfaro and Stephen Gilmore, editors. *Process Algebra and Probabilistic Methods*, LNCS 2165. Springer, September 2001. Proc. of the Joint International Workshop, PAPM-PROBMIV 2001, Aachen, Germany.

[7] R. Enders, T. Filkorn, and D. Taubner. Generating BDDs for symbolic model checking in CCS. *Distributed Computing*, 6(3):155–164, 1993.

[8] 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.

[9] H. Hermanns, J. Meyer-Kayser, and M. Siegle. Multi Terminal Binary Decision Diagrams to Represent and Analyse Continuous Time Markov Chains. In B. Plateau, W.J. Stewart, and M. Silva, editors, *3rd Int. Workshop on the Numerical Solution of Markov Chains*, pages 188–207. Prensas Universitarias de Zaragoza, 1999.

[10] W. Knottenbelt. Generalized Markovian Analysis of Timed Transition Systems, 1996. Master's Thesis, University of Cape Town.

[11] K. Lampka. Z-BDD-based State Graph Representation for Monolithic Model Descriptions. Technical Report 05/03, Universität Erlangen-Nürnberg, Institut für Informatik 7, 2003.

[12] K. Lampka and M. Siegle. Symbolic Composition within the Möbius Framework. In B. Wolfinger and K. Heidtmann, editors, *Proc. of the 2'nd MMB Workshop*, pages 63–74, September 2002. Technical Report of the Universität Hamburg, Department of Computer Science.

[13] S. Minato. Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems. In *Proc. of the 30th Design Automation Conference*, pages 272–277, Dallas (Texas), USA, June 1993. ACM Press.

[14] A. Miner and D. Parker. Symbolic Representations and Analysis of large State Spaces. In Ch. Baier, B. Haverkort, H. Hermanns, J.-P. Kartoen, and M. Siegle, editors, *Verification of Stochastic Systems, Proc. of the VOSS-Dagstuhl-Seminar*, December 2002. To appear.

[15] W.H. Sanders and J.F. Meyer. Reduced Base Model Construction Methods for Stochastic Activity Networks. *IEEE Journal on Selected Areas in Communications*, 9(1):25–36, January 1991.

[16] M. Siegle. Advances in model representation. In de Alfaro and Gilmore [6], pages 1–22. Proc. of the Joint International Workshop, PAPM-PROBMIV 2001, Aachen, Germany.

[17] M. Siegle. Behaviour analysis of communication systems: Compositional modelling, compact representation and analysis of performability properties. Habilitation thesis, Institut für Informatik, Friederich-Alexander-Universität Erlangen-Nürnberg, 2002.