Symbolic Composition within the Moebius Framework

Kai Lampka, Markus Siegle
Universität Erlangen-Nürnberg, Institut für Informatik

Abstract

This paper describes how to construct complex performability models in the context of the software tool Moebius, by hierarchically composing small submodels. In addition to Moebius' "Join" operator, a second composition operator "Sync" is introduced, and it is shown how both types of composition can be realised on the basis of symbolic, i.e. BDD-based data structures.

1 Introduction

Moebius [CCD+01] is a software tool for performability modelling which supports different model specification formalisms and allows the modeller to construct a complex overall model from small submodels. This paper describes how to extend Moebius in two directions:

We describe a technique for the compact representation of large state spaces, using Multi-terminal binary decision diagrams (MTBDD), which we apply to the Moebius modelling framework. Such a "symbolic" representation is based on a binary encoding of the labelled transition system (LTS), i.e. the states and state-to-state transitions, underlying a Moebius model. It is known that MTBDD-based state space representations can be extremely memory-efficient if the state space of a complex model is constructed in a compositional manner from small components.

In Moebius, submodels can be composed with the help of the operator *Join*, but in this paper we introduce another, new operator for composing submodels. This new composition operator is called *synchronisation* (*Sync*), it is well known in the world of process algebras and it constitutes a useful complementation of the *Join* operator already provided by Moebius. Fig. 1 shows the principle of *Join* and *Sync*, using two Stochastic Petri Net (SPN) submodels as an example. At the top of the figure, the SPN submodels are composed by a *Join*, which has the effect of superposing two places. At the bottom of the figure, the SPN submodels are composed by a *Sync*, which has the effect of superposing two transitions.

The MTBDD-based realisation of *Sync* has already been described in the literature, but in this paper we also describe an MTBDD-based realisation of *Join*,

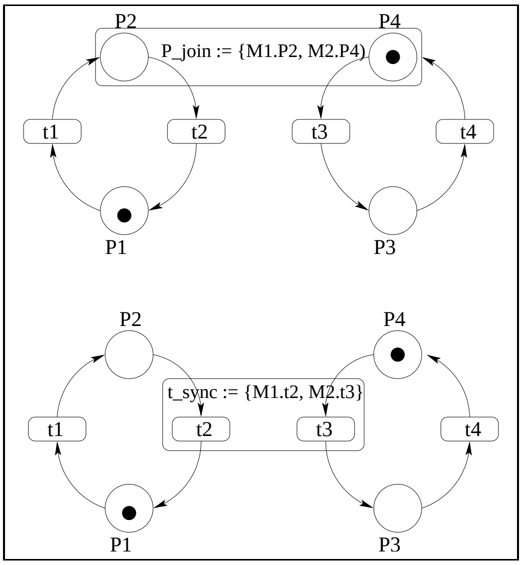

Figure 1: The operators *Join* and *Sync*.

thereby enabling a memory-efficient symbolic state space generation and representation within the Moebius modelling framework.

2 The Moebius modelling framework

Moebius is a software tool for performability evaluation of discrete event systems. Currently, Moebius supports the following three model specification formalisms [CCD+01, CS01]:

1. *Stochastic Activity Networks* (SAN) [SM91], an extension of *Generalised Stochastic Petri Nets* (GSPN),

2. *Performance Evaluation Process Algebra* (PEPA) [Hil94], a *Stochastic Process Algebra* (SPA), and

3. *Buckets and Balls* for modelling Markovian transition systems, as implemented, for instance, in the performance evaluation tools *Marca* [Ste91] and *DNAmaca* [Kno96].

A simple example model, consisting of a PEPA submodel *Producer* and a SAN submodel *Consumer*, is shown in Fig. 2. Within Moebius, submodels are mapped onto the *Abstract Functional Interface* (AFI), which is implemented in C++. Each place, bucket, process parameter or process counter of the specified submodels is hereby mapped onto a *state variable* (SV). During *state space* (SSp)

PEPA Submodel *Producer*:

Producer (Arrived) =
[Arrived < QCapacity]
$\Rightarrow$ (produce, λ).Producer(Arrived + 1);

SAN Submodel *Consumer*:

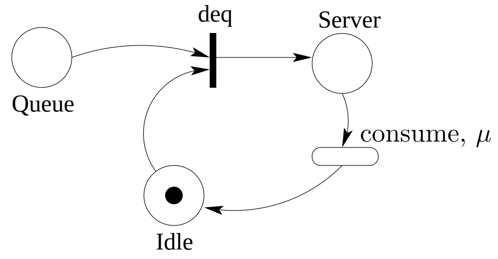

Figure 2: PC system consisting of a PEPA and a SAN submodel

exploration, a state of the modelled system is represented by a tuple of n SVs, which is called a *SSp vector* (SSpV) [Sow98].

In Moebius, models can be composed by applying a *Join*, which causes a sharing of designated SVs between the *atomic*[1] models [Doy97], as shown in Fig. 3 for the *Producer* and *Consumer* submodels (denoted as *PC* system in the following). On the level of the AFI, the *Join* results in a mapping of the explicitly shared SVs onto the same memory address, where in terms of PEPA models only SVs originating from process parameters can be shared. Such a mapping yields the effect that actions or transitions defined in different submodels may manipulate the shared but locally visible SVs and thus invisibly influence the behaviour of the partner models.

For the purpose of completeness, it should be briefly mentioned that within Moebius, the *Join* composition feature is augmented by the possibility of *replicating* designated submodels. During SSp generation, this explicit information of a composed model's symmetry is exploited for lumping states, leading to the generation of a monolithic but reduced overall SSp[2].

[1] In the present work, an *atomic* model is a model whose SSp is not the result of the composition of others. In process algebras, a *sequential* process is a process which does not contain the parallel composition operator.

[2] The term "monolithic" refers to the fact that the SSp of the composed model is not generated from the SSps of its submodels.

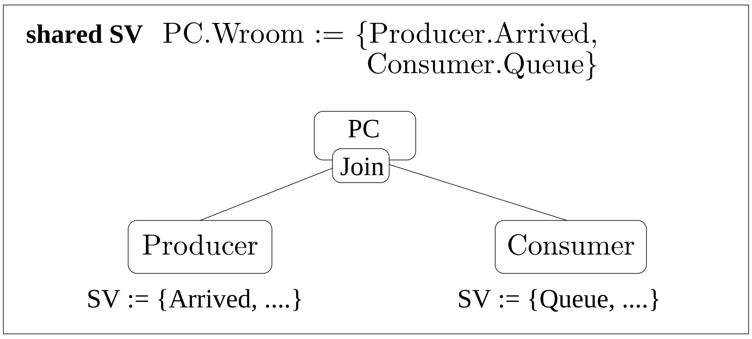

Figure 3: Hierarchic composition of the PC submodels via *Join*

3 Symbolic representation of state spaces

It is known that MTBDDs can be employed for representing large state spaces very compactly, provided that the overall state space is built in a compositional fashion [Sie01]. Before going into detail, we briefly recapitulate the basics.

MTBDDs [FMY97, BFG$^+$97]) are an extension of BDDs [Bry86] for the graph-based representation of pseudo-Boolean functions, i.e. functions of type $\mathbb{B}^n \mapsto \mathbb{R}$. 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 labelling 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 $n_a \geq \lceil \log_2 |\mathcal{A}ct| \rceil$ levels from the root are the variables a_i encoding the action, where $\mathcal{A}ct$ is the set of actions defined in a model whose SSp is to be encoded. On the remaining levels we have $2 * n_s \geq 2 * \lceil \log_2 |\mathcal{S}tates| \rceil$ variables encoding the source and target state of a transition. These $2{*}n_s$ variables (s_i) and (t_i) encoding the source and target states are ordered in an interleaved fashion, which yields the following overall variable ordering[3]:

$$\mathsf{a}_{n_a-1} \prec \ldots \prec \mathsf{a}_0 \prec \mathsf{s}_{n_s-1} \prec \mathsf{t}_{n_s-1} \prec \ldots \prec \mathsf{s}_0 \prec \mathsf{t}_0$$

The function represented by MTBDD M is denoted f_{M}. 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 [FMY97].

Fig. 4 shows an example stochastic LTS represented by an MTBDD, where a dashed (solid) line represents the Boolean values 0 (1) of the corresponding

[3]This interleaved ordering is the commonly accepted heuristics for obtaining small MTBDD sizes, see for instance [EFT93, FMY97, Sie01].

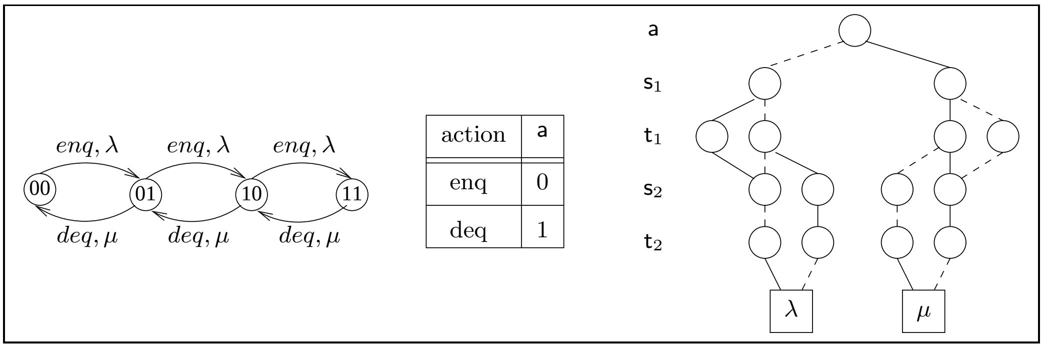

Figure 4: Stochastic LTS and corresponding MTBDD

Boolean variable. Note that MTBDD-based symbolic encodings of transition systems are also possible in the case where both Markovian and immediate transitions coexist [Sie02].

3.1 MTBDD-based representation of vector based SSps

The main idea of an MTBDD-based representation of vector-oriented SSps as realised by Moebius' AFI is that each SV will be encoded by its own Boolean vector, i.e. the current value of the associated SV (the marking of a specific place or bucket, the value of a process counter or parameter) is encoded in binary form. MTBDD-based SSp representation is only applicable if the model is finite and if upper bounds of all the SVs can be determined in advance[4]. Thus, the bound k_j on SV j yields the dimension $n_j \geq \lceil \log_2 k_j \rceil$ of the Boolean vectors $\vec{j}$ and $\vec{j}\,'$ which encode the value of that SV, where $\vec{j}$ represents the value before (*source SV*) and $\vec{j}\,'$ encodes the value after (*target SV*) a state transition (these were previously denoted as $\vec{\mathsf{s}}$ and $\vec{\mathsf{t}}$).

As an example, we consider the monolithic PC system shown in Fig. 5. For simplicity, the PEPA *Producer* process of Fig. 2 is here replaced by a SAN structure, and the system is now modelled as a flat SAN, where all SVs are 1-bounded, except the one representing place *Queue* which is bounded by $QCapacity = 3$. After the dimensions of the Boolean vectors are determined (see Table 1), one can explore the SSp of the whole model in one step (monolithically) and encode the transition system as an MTBDD [Dav01].

[4]Such bounds can be computed, for example, with the help of P-invariant analysis.

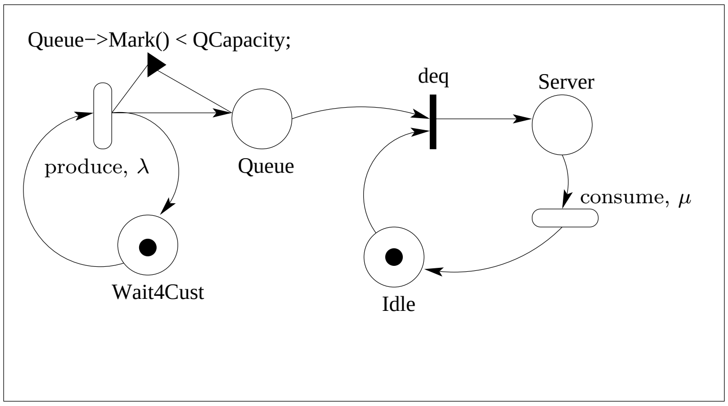

Figure 5: PC system described as a monolithic SAN

	Queue	Wait4Cust	Idle	Server	transitions
bound k	3	1	1	1	3
$\lceil \log_2 k \rceil$	2	1	1	1	2
Boolean vector	$\vec{q},\ \vec{q}\,'$	$\vec{w},\ \vec{w}\,'$	$\vec{i},\ \vec{i}\,'$	$\vec{s},\ \vec{s}\,'$	$\vec{a}$

Table 1: Encoding scheme of SSp of the PC-model

3.2 MTBDD-based Synchronisation

When composing two submodels P_1 and P_2 in parallel, a subset of the actions may have to be executed jointly by both partners. In the context of stochastic process algebras this is called synchronisation, or *Sync* for short. As a result, the SSp T of the composed process is a subset of the Cartesian product of the SSps T_i of the submodels, denoted as product SSp (PSSp), i.e.:

$$P = P_1 < S > P_2 \quad \Rightarrow \quad T \subseteq T_1 \times T_2.$$

As an example consider Fig. 6. There the rate of action a in submodel P_3, which results from a synchronisation of P_1 and P_2, is given by $\varphi(\lambda,\ \mu)$. In case of the SPA TIPP [HHK+00], the rate of a synchronised Markovian transition is given by $\varphi(\lambda,\ \mu) = \ \lambda \cdot \mu$. However, other synchronisation policies are also possible. In process P_4, which is the unsynchronised composition of P_1 and P_2, we can observe the interleaving of two a actions.

On the MTBDD level, the operator Sync can be realised as follows [Sie02]: Let P_1 and P_2 be two processes to be synchronised over a set S of actions. The corresponding SLTSs are encoded as MTBDD M_i over the Boolean vectors $(\vec{a}, \vec{s}_i, \vec{t}_i)$, where $i \in \{1, 2\}$. The set S of synchronising transitions is encoded as BDD S over the Boolean vector $\vec{a}$. Furthermore, $Stab_i$ is a BDD which encodes stability

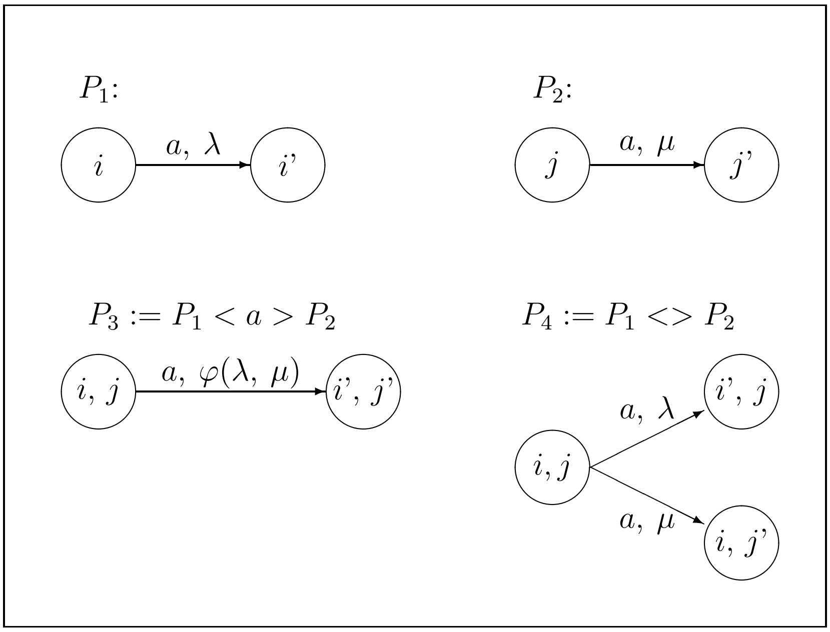

Figure 6: Composition of processes, synchronisation and interleaving

of submodel i, i.e. the fact that the source state equals the target state. The MTBDD M which encodes the SLTS T is then given by:

$$\boxed{\begin{array}{lll} \mathsf{M} := & & (\mathsf{M}_1 \cdot \mathsf{S}) \cdot (\mathsf{M}_2 \cdot \mathsf{S}) \\ & + & \mathsf{M}_1 \cdot (1 - \mathsf{S}) \cdot Stab_2 \\ & + & \mathsf{M}_2 \cdot (1 - \mathsf{S}) \cdot Stab_1 \end{array}}$$

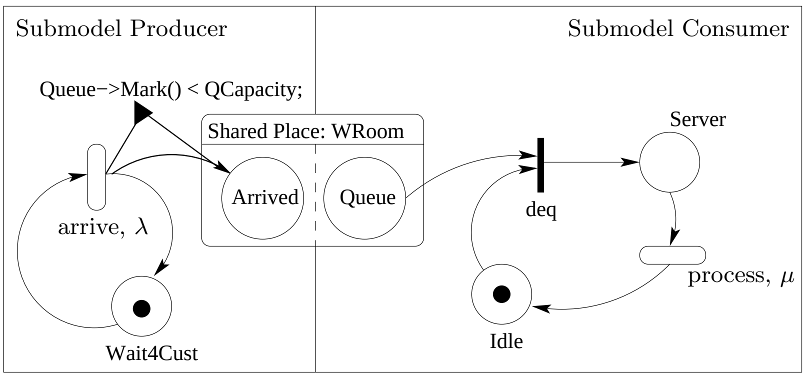

Figure 7: PC system modelled by two SAN submodels

4 Extending Moebius

In the following, we describe how a *synchronisation* of timed activities can be introduced into Moebius, where in the following "activity" refers to a timed transition of a *SAN* or *Bucket and Balls* model, or a timed action of a *PEPA* model. After introducing *Sync* into Moebius, we also develop an MTBDD-based Join. Together, *Sync* and *Join* are basic preconditions for an effective use of an MTBDD-based SSp representation within the Moebius modelling framework.

4.1 Introducing Synchronisation into Moebius

Referring to the vector layout of the monolithic SSp as established by the AFI, the main idea of a synchronisation is the *conjunction* of the entrance conditions and of the consequences of the jointly executed activities (for exemplification the reader may refer to Fig. 6):

Conjunction of entrance conditions of activity a:
The timed activity a needs to be enabled in all submodels which are required to participate in its execution, i.e. all respective places or buckets in the submodel(s) must contain the required amount of tokens. In case of PEPA submodels the conditional constructs (guards) of a need to evaluate to true.

Conjunction of consequences, when activity a is executed:
The monolithic PSSp of the composed model consists of arbitrary PSSp vectors of the form $(\vec{j_1}, \vec{j_2})$. In the following, let $S \subseteq \mathcal{A}ct_1 \cap \mathcal{A}ct_2$ be the set of activities to be jointly executed:

1. **Interleaved** execution of $a \notin S$:

$$\frac{\vec{j_1} \xrightarrow{a,\lambda} \vec{j_1}'}{(\vec{j_1}, \vec{j_2}) \xrightarrow{a,\lambda} (\vec{j_1}', \vec{j_2})} \quad \boxed{a \in \mathcal{A}ct_1}$$

$$\frac{\vec{j_2} \xrightarrow{a,\mu} \vec{j_2}'}{(\vec{j_1}, \vec{j_2}) \xrightarrow{a,\mu} (\vec{j_1}, \vec{j_2}')} \quad \boxed{a \in \mathcal{A}ct_2}$$

2. **Synchronised** execution of $a \in S$:

$$\frac{\vec{j_1} \xrightarrow{a,\lambda} \vec{j_1}' \wedge \vec{j_2} \xrightarrow{a,\mu} \vec{j_2}'}{(\vec{j_1}, \vec{j_2}) \xrightarrow{a,\varphi(\lambda,\mu)} (\vec{j_1}', \vec{j_2}')} \quad \boxed{a \in \mathcal{A}ct_1, a \in \mathcal{A}ct_2}$$

(Concerning the rate $\varphi(\lambda, \mu)$ of a synchronised transition, we follow the TIPP policy as described in Section 3.2.) This approach can be extended to the synchronisation of timeless activities and to the case of more than two submodels to be synchronised.

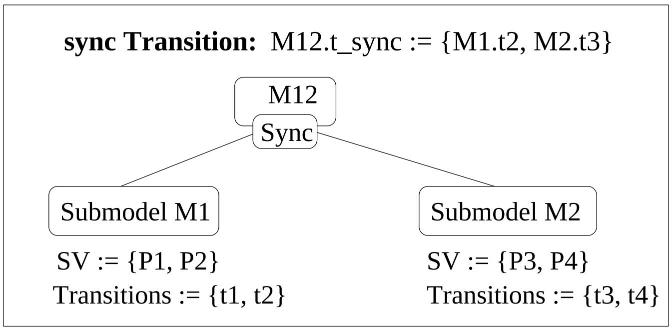

Figure 8: Hierarchic composition of the models M1 and M2 via *Synchronisation*

We now refer to the example shown in Fig. 1, where in the lower part a superposition of two transitions t2 and t3, defined in two different SANs M1 and M2, is illustrated. The structure of the composed model M12 could then be described as illustrated in Fig. 8[5]. Once Moebius includes a *Sync* operator, one can apply a compositional SSp construction, based on local SSps represented by MTBDDs and their parallel composition as already described in Section 3.2.

4.2 MTBDD-based Join

Consider the PC system given in Fig. 7, composed in the same manner as shown in Fig. 3, however for simplicity modelled by two SAN submodels (which is transparent on the level of composition). The respective encoding scheme is then shown in Table 2. The main idea of MTBDD-based *Join* is to compose the submodel SSps over the values of the shared places (and not over transition labels, as done before). Each shared SV in the different submodels needs to be referenced by the same Boolean vector in the MTBDDs to be joined. Therefore the Boolean vectors representing the SVs in a submodel are mapped on the following Boolean vectors (see Table 3):

- $\vec{s}^{\,0}$, representing the shared source SVs,
- $\vec{s}^{\,i}$, the local source SVs of submodel i,
- $\vec{t}^{\,0}$, counterpart of vector $\vec{s}^{\,0}$, but representing the target SVs,
- $\vec{t}^{\,i}$, counterpart of vector $\vec{s}^{\,i}$, representing the target SVs.

The symbolic SSp representation of a composed system, which is the *Join* of the SSps of the two submodels which are already represented as MTBDDs, can be constructed as follows:

[5] Note that this approach does not yet consider priorities of transitions.

Submodel *Producer*

transitions			Arrived		Wait4Cust	Arrived'		Wait4Cust'	
produce	1	0	0	0	1	0	1	1	λ
			0	1	1	1	0	1	λ
			1	0	1	1	1	1	λ
Vars	a_0	a_1	ar_0	ar_1	w_0	a'_0	a'_1	w'_0	

Submodel *Consumer*

transitions			Source state				target state				rate/ weight
			Queue		Server	Idle	Queue'		Server'	Idle'	
deq	0	0	1	1	0	1	1	0	1	0	1
			1	0	0	1	0	1	1	0	1
			0	1	0	1	0	0	1	0	1
consume	0	1	1	1	1	0	1	1	0	1	μ
			1	0	1	0	1	0	0	1	μ
			0	1	1	0	0	1	0	1	μ
			0	0	1	0	0	0	0	1	μ
Vars	a_0	a_1	q_0	q_1	s_0	i_0	q'_0	q'_1	s'_0	i'_0	

Table 2: Binary encoding of the transition relations for PC submodels

Submodel	Source state					target state				
Consumer	q_0	q_1	s_0	i_0		q'_0	q'_1	s'_0	i'_0	
Producer	ar_0	ar_1			w_0	ar'_0	ar'_1			w'_0
	s^0_0	s^0_1	s^1_1	s^1_2	s^2_1	t^0_0	t^0_1	t^1_1	t^1_2	t^2_1
	$\vec{s}^{\,0}$		$\vec{s}^{\,1}$		$\vec{s}^{\,2}$	$\vec{t}^{\,0}$		$\vec{t}^{\,1}$		$\vec{t}^{\,2}$

Table 3: Mapping of the Boolean variables onto four Boolean vectors

Let P_1 and P_2 be submodels, sharing a set of places, buckets or process parameters, i.e. sharing a set of SVs. The corresponding SLTS are encoded as MTBDD M_i over the Boolean vectors $(\vec{a}, \vec{s}^{\,0}, \vec{s}^{\,i}, \vec{t}^{\,0}, \vec{t}^{\,i}),\ i \in \{1, 2\}$. The LTS M encoding the composed system can be constructed as follows, (where the BDDs $\widetilde{Stab}_i$ express the fact that the Boolean variables of the "passive" partner remain stable, i.e. that the vectors $\vec{s}^{\,i}$ and $\vec{t}^{\,i}$ are identical):

$$\boxed{\mathsf{M} \ := \ \mathsf{M}_1 \cdot \widetilde{Stab}_2 + \mathsf{M}_2 \cdot \widetilde{Stab}_1}$$

This approach, as well as the one for the MTBDD-based synchronisation (Section 3.2) may require a symbolic reachability analysis, which is due to the encoding of unreachable source states [Sie02].

5 Conclusion

As discussed in this paper, Moebius can be extended by a composition operator *Sync*, which realises a superposing of transitions defined in different (possibly heterogenous) submodels. We can employ an MTBDD-based realisation of *Sync*, as had already been described in the literature. As a new feature, we have developed the MTBBD-based realisation of Moebius' composition operator *Join*. Thus, a memory-efficient symbolic SSp representation (based on MTBDDs) can be put into practice within the Moebius modelling framework. Apart from implementing the approach described in this paper, our future work will also address open questions concerning the MTBDD-based realisation of Moebius' *replicate* feature (see Section 2) and the handling of reward variables.

References

[BFG+97] 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.

[Bry86] R.E. Bryant. Graph-based Algorithms for Boolean Function Manipulation. *IEEE ToC*, C-35(8):677–691, August 1986.

[CCD+01] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. Webster. The Moebius Modeling Tool. In de Alfaro and Gilmore [dAG01], pages 241–250.

[CS01] Graham Clark and William H. Sanders. Implementing a Stochastic Process Algebra within the Moebius Modeling Framework. In de Alfaro and Gilmore [dAG01], pages 200–215.

[dAG01] Luca de Alfaro and Stephen Gilmore, editors. *Process Algebra and Probabilistic Methods*. Springer, LNCS 2165, September 2001.

[Dav01] Ian Davies. Symbolic Techniques for the Performance Analysis of Generalized Stochastic Petri Nets, Master Thesis, University of Cape Town (South Africa), 2001.

[Doy97] Jay M. Doyle. Abstract Model Specification using the Moebius Modelling Tool, Master Thesis, University of Illinois at Urbana-Champaign (Illinois, USA), 1997.

[EFT93] R. Enders, T. Filkorn, and D. Taubner. Generating BDDs for symbolic model checking in CCS. *Distributed Computing*, (6):155–164, 1993.

[FMY97] 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.

[HHK+00] H. Hermanns, U. Herzog, U. Klehmet, V. Mertsiotakis, and M. Siegle. Compositional performance modelling with the TIPPtool. *Performance Evaluation*, 39(1-4):5–35, January 2000.

[Hil94] J. Hillston. A Compositional Approach to Performance Modelling, Ph.D. Thesis, University of Edinburgh (United Kingdom), 1994.

[Kno96] W. Knottenbelt. Generalized Markovian Analysis of Timed Transition Systems. Master Thesis, University of Cape Town (South Africa), 1996.

[Sie01] M. Siegle. Advances in model representation. In de Alfaro and Gilmore [dAG01], pages 1–22.

[Sie02] Markus Siegle. Behaviour analysis of communication systems: Compositional modelling, compact representation and analysis of performability properties. Habilitationsschrift, Institut für Informatik, Friederich-Alexander-Universität Erlangen-Nürnberg, 2002.

[SM91] 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.

[Sow98] John Marks Sowder. State-Space Generation Techniques in the Moebius Framework, Master Thesis, University of Illinois at Urbana-Champaign (Illinois, USA), 1998.

[Ste91] W.J. Stewart. MARCA: Markov Chain Analyzer, A Software Package for Markov Modeling. In W.J. Stewart, editor, *Numerical Solution of Markov Chains*. Marcel Dekker, 1991.