Bound-Preserving Composition for Markov Reward Models*

David Daly
IBM T.J. Watson Research Center
Yorktown Heights, NY, 10958, U.S.A.
dmdaly@us.ibm.com

Peter Buchholz
Informatik IV, Universität Dortmund
D-44221 Dortmund, Germany
peter.buchholz@udo.edu

William H. Sanders
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 W. Main St., Urbana, IL, U.S.A.
whs@crhc.uiuc.edu

Abstract

Stochastic orders can be applied to Markov reward models and used to aggregate models, while introducing a bounded error. Aggregation reduces the number of states in a model, mitigating the effect of the state-space explosion and enabling the wider use of Markov reward models. Existing aggregation techniques based upon stochastic orders are limited by a combination of strong requirements on the structure of the model, and complexity in determining the stochastic order and generating the aggregated model. We develop a set of general conditions in which models can be analyzed and aggregated compositionally, dramatically lowering the complexity of the aggregation and solution of the model. When these conditions are combined with a recently developed general stochastic order for Markov reward models, significantly larger models can be solved than was previously possible for a large class of models.

1 Introduction

Markov reward models are an important model type for performance and dependability analysis of discrete event systems. Although transient and stationary analysis of continuous time Markov chains (CTMCs) and discrete time Markov chains (DTMCs) is theoretically easy and a large number of numerical methods exist for this purpose [19], practical problems arise due to the so-called *state space explosion problem*. Thus, analysis of large CTMCs or DTMCs is still a challenging problem. For state-space sizes beyond numerically solvable sizes, simulation, approximations, or bounding techniques can be used. Simulation and approximation techniques suffer from the problem that an unknown error can make them hard to use in cases where rare events have to be analyzed. Bounding methods are more attractive, since the quality of bounds can be easily evaluated.

Several bounding methods exist. For stationary analysis, the method of Courtois and Semal [7] is very prominent and has led to a couple of extended methods [15, 13, 3]. However, those approaches can only be applied for stationary and not transient analysis, and often can only be applied for specific Markov chains.

An alternative bounding approach uses the more general concepts of stochastic ordering and comparison [14]. In stochastic comparisons, random variables are compared and conditions are established that one random variable is larger than another. In the context of CTMCs or DTMCs with rewards, this implies that one process yields a larger reward than another. Using the concepts of strong stochastic ordering and stochastic monotonicity, orderings between states of a Markov chain and different Markov chains can be established [12, 20]. The approach can be used to compute bounds for various measures of Markov models [10, 16] as shown recently.

However, the restrictions for strong stochastic orderings as applied in the mentioned papers are very strict. That is, a state with a larger index has to be stochastically larger than a state with a smaller index. This implies that for many models, the resulting bounds are very loose. Another deficit of the available approaches is that they all work on a complete CTMC or DTMC, possibly by exploiting some of its structure. However, models can also be specified as a number of

*This material is based upon work supported in part by the National Science Foundation under Grant Nos. CCR-0311616 and INT-0233490 and by Pioneer Hi-Bred International, Inc.

smaller parts denoted as components, and a specification of how the components interact, called composition. It turns out that for large state spaces, techniques that apply aggregation or state-space reduction compositionally to the state spaces of some components before they are composed with the rest of the system are especially successful. Take as an example the work on stochastic bisimulation in stochastic process algebras or automata networks [11, 2], in which an exact aggregation of components can be applied before the complete CTMC is built. For stochastic orderings, we are aware of only one approach that applies the technique in a compositional setting. In [20] it is shown that for a stochastic automata network without synchronizations, stochastic ordering in one component is preserved after the component is combined with the rest of the system. However, without synchronization, the modeling power of stochastic automata is very restricted, and the interesting cases are left out.

In this paper we extend the bounding approaches based on stochastic orderings. In [8, 9] we developed a new ordering relation for the states of a DTMC/CTMC or between two DTMCs/CTMCs that generalizes the strong stochastic ordering relation used in the above-mentioned papers. This relation allows us to generate, for some Markov chains, aggregated Markov chains that bound the transient and stationary results of the original Markov chains. In this paper, we show that the approach can be applied to the components of a large Markov model and that the ordering is preserved if certain conditions of the composition are observed. In contrast to that described in [20], our approach allows synchronization between components but requires some restrictions on the component behavior (see theorems 4.1-**??**) . Since we use an ordering-relation that is a generalization of strong stochastic ordering, the results we present generalize the composition results of [20] as well.

The outline of the paper is as follows. In the next section we briefly review our ordering relation and the corresponding aggregation of Markov models. Section 3 introduces the composition of components of a Markov model. In Section 4, the preservation of state orderings for different kinds of composition, which is the main result of this paper, is shown, and we briefly present a numerical example in Section 5.

2 A Preorder for Markov Reward Models

First we review the key definitions and theorems from [8, 9]. Proofs of the theorems are omitted and may be found in [9], which is available on the Web.

The preorder we use for the comparison of states is defined on some Markov process X_t with a finite or countable state space $\mathcal{S}$, initial distribution p_0 ($p_0 : \mathcal{S} \to \mathbb{R}$ and $\sum_{i\in\mathcal{S}} p_0(i) = 1.0$), and additional reward function r ($r : \mathcal{S} \to \mathbb{R}_+$), which assigns a non-negative reward to the states. We first derive the results for DTMCs and then show how they can be transferred to CTMCs using uniformization [19].

Let $\mathbf{P}$ be the transition matrix of the DTMC. $\mathbf{P}$ is a stochastic matrix of dimension $|\mathcal{S}|$ where $|\mathcal{S}|$ is the size of the state space and $\mathbf{P}(i, j)$ is the transition probability from state i to j. Similarly, the initial distribution and the reward function can be described as an $|\mathcal{S}|$-dimensional row vector $\mathbf{p}_0$ and an $|\mathcal{S}|$-dimensional column vector $\mathbf{r}$, respectively. The vector of all ones and the vector with 1 at index x are represented by $\mathbf{e}$ and $\mathbf{e}_x$ respectively. We use the notation $DM = (\mathcal{S}, \mathbf{P}, \mathbf{p}_0, \mathbf{r})$ for a DTMC with reward; the random variable $R_k(\mathbf{e}_x)$ denotes the reward after k steps given that the model starts in state x, and $R_k(\mathbf{p}_0)$ is the reward after k steps starting with the initial distribution $\mathbf{p}_0$.

The following two definitions from [9] extend the definition of stochastic order to states in a Markov chain, and to complete DTMCs. The definitions say that two states or models are stochastically ordered if the states or models lead to stochastically ordered results. General results about different stochastic orderings can be found in [14].

Definition 2.1 *Let $DM = (\mathcal{S}, \mathbf{P}, \mathbf{p}_0, \mathbf{r})$ be a DTMC with rewards and $x, y \in \mathcal{S}$. We say $x \geq_{st} y$ iff $R_k(\mathbf{e}_x) \geq_{st} R_k(\mathbf{e}_y)$ for all $k = 0, 1, \ldots$*

Definition 2.2 *Let $DM^{(i)} = (\mathcal{S}^{(i)}, \mathbf{P}^{(i)}, \mathbf{p}_0^{(i)}, \mathbf{r}^{(i)})$ ($i = 1, 2$) be two DTMCs. We say $DM^{(1)} \geq_{st} DM^{(2)}$ iff $R_k^{(1)}(\mathbf{p}_0^{(1)}) \geq_{st} R_k^{(2)}(\mathbf{p}_0^{(1)})$ for all $k = 0, 1, \ldots$*

The following two theorems use the well-known uniformization approach to transform a CTMC into a DTMC [19] and show that a stochastic ordering on the states of a uniformized CTMC, or between uniformized CTMCs, implies a stochastic ordering on the states of the CTMC or between the CTMCs, and therefore that the key results also hold for CTMCs.

Theorem 2.1 *Let $CM = (\mathcal{S}, \mathbf{Q}, \mathbf{p}_0, \mathbf{r})$ be a uniformizable CTMC and $DM_\alpha = (\mathcal{S}, \mathbf{Q}/\alpha + \mathbf{I}, \mathbf{p}_0, \mathbf{r})$ be the uniformized DTMC for uniformization rate α. If for some α with $\max_{i\in\mathcal{S}} |\mathbf{Q}(i, i)| \leq \alpha < \infty$: $x \geq_{st} y$ in DM_α, then $x \geq_{st} y$ in CM.*

Theorem 2.2 *Let $CM^{(i)} = (\mathcal{S}^{(i)}, \mathbf{Q}^{(i)}, \mathbf{p}_0^{(i)}, \mathbf{r}^{(i)})$ ($i = 1, 2$) be two CTMCs and $DM^{(i)} = (\mathcal{S}^{(i)}, \mathbf{Q}^{(i)}/\alpha + \mathbf{I}, \mathbf{p}_0^{(i)}, \mathbf{r}^{(i)})$ the corresponding uniformized DTMCs for some appropriate uniformization rate α with $\max_{i=1,2}(\max_{x\in\mathcal{S}^{(i)}}(\mathbf{Q}^{(i)}(x, x)|)) \leq \alpha < \infty$. If $DM^{(1)} \geq_{st} DM^{(2)}$ then $CM^{(1)} \geq_{st} CM^{(2)}$.*

Let $\mathbf{P}(x\bullet)$ denote row x of matrix $\mathbf{P}$. A preorder $\succeq$ is a reflexive and transitive relation among the states from $\mathcal{S}$. We use the notation $i \succeq j$ to indicate that i and j are related

according to preorder $\succeq$. Before we define a preorder relation that implies stochastic ordering (i.e., $x \succeq y \Rightarrow x \geq_{st} y$), we recapitulate the definition of *decomposition*.

Definition 2.3 *A decomposition ϕ of a non-negative vector $\mathbf{a}$ over finite or countable state space $\mathcal{S}$ is a set of vectors $(\mathbf{a}_1, \ldots, \mathbf{a}_{K_\phi})$ with $\mathbf{a}_k \geq \mathbf{0}$ and $\sum_{k=1}^{K_\phi} \mathbf{a}_k = \mathbf{a}$. K_ϕ is the size of the decomposition, which can be infinite.*

Definition 2.4 *Let $\mathcal{S}$ be a state space, $\succeq$ be a preorder on the states of $\mathcal{S}$, and $\mathbf{a}, \mathbf{b}$ be two distributions over the state space $\mathcal{S}$; then, $\mathbf{a} \succeq \mathbf{b}$ iff a decomposition ϕ of size $|\mathcal{S}|$ exists such that for every $x \in \mathcal{S}$: $\sum_{y \succeq x} \mathbf{a}_x(y) \geq \mathbf{b}(x)$.*

Intuitively, this definition says that $\mathbf{a} \succeq \mathbf{b}$ if every entry in $\mathbf{b}$ can be replaced by some collection of entries in $\mathbf{a}$, all of which are greater than the entry in $\mathbf{b}$ according to $\succeq$.

The previous definition is used to define preorder relations that order states according to their contribution to the reward measures, and order models based on their reward solutions.

Definition 2.5 *A preorder $\succeq$ on the state space of a DTMC $DM = (\mathcal{S}, \mathbf{P}, \mathbf{p}_0, \mathbf{r})$ is reward-preserving, iff $x \succeq y$ implies*

1. $\mathbf{r}(x) \geq \mathbf{r}(y)$ *and*
2. $\mathbf{P}(x\bullet) \succeq \mathbf{P}(y\bullet)$.

Or, in other words, a preorder is reward-preserving if the rewards are greater in state x than in state y, and x has larger next states than does y.

Theorem 2.3 *If for two states $x, y \in \mathcal{S}$ $x \succeq y$ and $\succeq$ is reward-preserving, then $x \geq_{st} y$.*

Definition 2.6 *Let $DM^{(i)} = (\mathcal{S}^{(i)}, \mathbf{P}^{(i)}, \mathbf{p}_0^{(i)}, \mathbf{r}^{(i)})$ $(i = 1, 2)$ with $\mathcal{S}^{(1)} \cap \mathcal{S}^{(2)} = \emptyset$; then, the union of the two models is $DM^{(12)} = DM^{(1)} \cup DM^{(2)} = (\mathcal{S}^{(12)}, \mathbf{P}^{(12)}, \mathbf{p}_0^{(12)}, \mathbf{r}^{(12)})$ with state space defined as $\mathcal{S}^{(12)} = \mathcal{S}^{(1)} \cup \mathcal{S}^{(2)}$,*

$$\mathbf{P}^{(12)} = \begin{pmatrix} \mathbf{P}^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}^{(2)} \end{pmatrix},$$

$$\mathbf{p}_0^{(12)} = 0.5\left(\mathbf{p}_0^{(1)}, \mathbf{p}_0^{(2)}\right) \text{ and } \quad \mathbf{r}^{(12)} = \left(\mathbf{r}^{(1)}, \mathbf{r}^{(2)}\right)$$

Theorem 2.4 *Let $DM^{(i)} = (\mathcal{S}^{(i)}, \mathbf{P}^{(i)}, \mathbf{p}_0^{(i)}, \mathbf{r}^{(i)})$ $(i = 1, 2)$ and $\succeq$ be a reward-preserving preorder on the union of both DTMCs; then, $DM^{(1)} \geq_{st} DM^{(2)}$ if $\left(\mathbf{p}_0^{(1)}, \mathbf{0}\right) \succeq \left(\mathbf{0}, \mathbf{p}_0^{(2)}\right)$.*

A basic algorithm for generating a reward-preserving preorder ($\succeq_\sim$) by stepwise refinement for any model is given in [9]. It is also shown in [9] that the generated preorder is the coarsest reward-preserving preorder.

Theorem 2.5 *If $\succeq_*$ is a reward-preserving preorder on the state space of a DM and the preorder $\succeq_\sim$ exists, then $x \succeq_* y \Rightarrow x \succeq_\sim y$.*

Finally, it is shown in [9] how the preorder may be used to generate bounding aggregates that can be solved for results that stochastically bound the results for the unaggregated model.

Definition 2.7 *Let $DM = (\mathcal{S}, \mathbf{P}, \mathbf{p}_0, \mathbf{r})$ and $\Pi = \{\mathcal{S}_1, \ldots, \mathcal{S}_J\}$ be a partition of the state space.*

- *Π contains least states, iff for all $\mathcal{S}_j$ some $x \in \mathcal{S}_j$ exists such that $x \preceq_\sim y$ for all $y \in \mathcal{S}_j$,*
- *Π contains greatest states, iff for all $\mathcal{S}_j$ some $x \in \mathcal{S}_j$ exists such that $x \succeq_\sim y$ for all $y \in \mathcal{S}_j$.*

Definition 2.8 *Let $DM = (\mathcal{S}, \mathbf{P}, \mathbf{p}_0, \mathbf{r})$ be a DTMC with rewards and $\Pi = \{\mathcal{S}_1, \ldots, \mathcal{S}_J\}$ be a partition of its state space. The following aggregated processes can be defined: $\tilde{\mathcal{S}} = \{1, \ldots, J\}$, $\tilde{\mathbf{P}} \in \mathbb{R}^{J \times J}$, $\tilde{\mathbf{p}}_0 \in \mathbb{R}^J$ and $\mathbf{r} \in \mathbb{R}^J$.*

1. *If Π contains least states and x_j^- is the least state of $\mathcal{S}_j$, then $DM_\Pi^- = (\tilde{\mathcal{S}}, \tilde{\mathbf{P}}, \tilde{\mathbf{p}}_0, \tilde{\mathbf{r}})$, where $\tilde{\mathbf{p}}_0(j) = \sum_{y \in \mathcal{S}_j} \mathbf{p}_0(y)$, $\tilde{\mathbf{r}}(j) = \mathbf{r}(x_j^-)$, and $\tilde{\mathbf{P}}(j, l) = \sum_{y \in \mathcal{S}_l} \mathbf{P}(x_j^-, y)$.*
2. *If Π contains greatest states and x_j^+ is the greatest state $\mathcal{S}_j$, then $DM_\Pi^+ = (\tilde{\mathcal{S}}, \tilde{\mathbf{P}}, \tilde{\mathbf{p}}_0, \tilde{\mathbf{r}})$ where $\tilde{\mathbf{p}}_0(j) = \sum_{y \in \mathcal{S}_j} \mathbf{p}_0(y)$, $\tilde{\mathbf{r}}(j) = \mathbf{r}(x_j^+)$ and $\tilde{\mathbf{P}}(j, l) = \sum_{y \in \mathcal{S}_l} \mathbf{P}(x_j^+, y)$.*

The significance of a partition having greatest or least states is shown in the next theorem. A partition with greatest or least states can be used to generate an aggregated model that stochastically bounds the original model.

Theorem 2.6 *The relation $DM_{\Pi_u}^+ \geq_{st} DM \geq_{st} DM_{\Pi_l}^-$ holds.*

A limiting factor in computing the preorder and using it to create bounding aggregates is the size of the state space. Previous analysis and experiments have shown the order of computation to be polynomial in the number of states [8], so that computing the preorder on complete state spaces is limited by the state-space explosion. Thus, from an application point of view, it is important to use the aggregation compositionally by first aggregating components and then combining the aggregated components with the environment. However, such an approach only works if the preorder relation is

preserved by the composition. For the rest of this paper, we derive conditions under which our preorder is preserved. To do so, we first introduce composition of Markov models.

3 Composition Operations

Composition of Markov models has been defined in different frameworks. We use here an approach which applies Kronecker products and originates in the work on stochastic automata networks [17] and asynchronous composition of queuing networks [4]. For the presentation of composition we first present a small running example; then we define Kronecker operations as the basis for composition at the state level; and lastly we introduce the composition of components and distinguish between asynchronous and synchronous composition. We present composition here for continuous time models, but it could as well be applied for discrete time models [1, 18].

3.1 A Running Example

To illustrate the composition, we use a small, running example. The example shown in Figure 1 is of a fault-tolerant computing system that processes requests. There are two parts to the system: a front-end request-processing system (top of figure) and a fault-tolerant back-end (bottom part of figure). The front-end processes requests according to a Cox-2 distribution provided that the back-end is operational, while requests arrive at the system with Cox-2 distributed inter-arrival times. If the back-end is not operational, no requests are processed by the front-end.

The back-end consists of two sets of units. The units fail with a rate dependent on the queue length of the front-end, and the back-end is considered operational so long as one of each unit type is operational. The two units can be repaired independently, but may also take part in a shared repair process. The shared repair process repairs one of each unit, and the rate of the shared process is a non-decreasing function of the number of operational units. (A unit will participate in the shared repair process even if it has no failed units.) We use the example to demonstrate each type of composition, and in Section 5 we solve the model using aggregation.

3.2 Kronecker Product and Sum

The composition of systems relies on the composition of Markov chains. The resulting structures are well-established, and it is known that the generator matrix of the composed model can be described as a sum of Kronecker products of small component matrices, which is a useful structure in proving compositional results and realizing algorithms [17, 18, 4, 1].

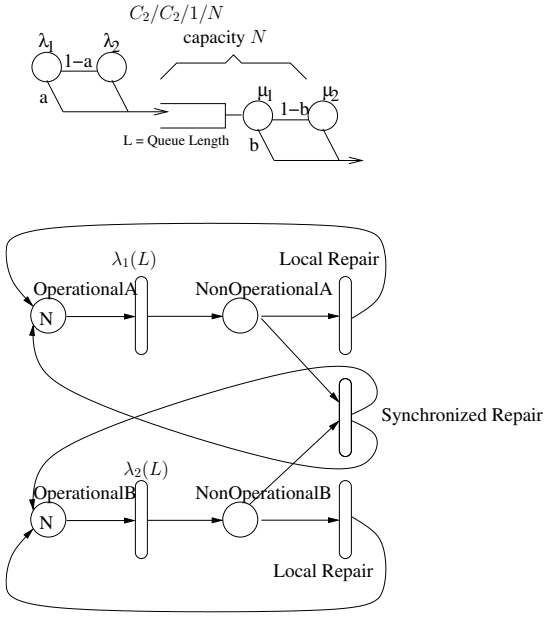

Figure 1. A fault-tolerant computing system that processes requests and has multiple redundant units that may fail.

The Kronecker product is represented by the symbol $\bigotimes$, and the Kronecker sum by the symbol $\bigoplus$. The Kronecker product and sum are functions of matrices, and are defined as

$$\mathbf{A} \bigotimes \mathbf{B} = \begin{pmatrix} a_{1,1}\mathbf{B} & \dots & a_{1,n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m,1}\mathbf{B} & \dots & a_{m,n}\mathbf{B} \end{pmatrix}$$

and

$$\mathbf{A} \bigoplus \mathbf{B} = \mathbf{A} \bigotimes \mathbf{I} + \mathbf{I} \bigotimes \mathbf{B}$$

Since Kronecker products are associative, they can be naturally extended to more than two matrices.

3.3 Composition of Components

We consider the composition of two components. This operation can be repeated to compose more than two components. Composition is performed over a set of transitions $\mathcal{K} = \{1, \dots, K\}$. Let λ_k ($k \in \mathcal{K}$) be the basic transition rate of transition k.

Definition 3.1 *Component i is characterized by $(\mathcal{S}^{(i)}, (\mathbf{Q}_0^{(i)}, \mathbf{P}_1^{(i)}, \dots, \mathbf{P}_K^{(i)}), \mathbf{p}_0^{(i)}, \mathbf{r}^{(i)})$, where $\mathcal{S}^{(i)}$ is the state space of size $n^{(i)}$, $\mathbf{Q}_0^{(i)}$ ($\mathbf{0} \leq \mathbf{Q}_0^{(i)}$) is the rate matrix of local transitions, $\mathbf{P}_k^{(i)}$ ($\mathbf{0} \leq \mathbf{P}_k^{(i)}$, $\mathbf{P}_k^{(i)}\mathbf{e}^T \leq 1$ and $k \in \mathcal{K}$) are the weight matrices of synchronized*

transitions (the probability that component i will offer the synchronized transition to the environment), $\mathbf{p}_0^{(i)}$ *is the initial distribution, and* $\mathbf{r}^{(i)}$ *is the reward vector.*

The descriptor matrix of a model composed of components 1 and 2 is given by

$$\mathbf{Q} = \begin{array}{c} \mathbf{Q}_0^{(1)} \bigoplus \mathbf{Q}_0^{(2)} - \mathbf{D}_0^{(1)} \bigoplus \mathbf{D}_0^{(2)} + \\ \sum_{k\in\mathcal{K}} \lambda_k (\mathbf{P}_k^{(1)} \bigotimes \mathbf{P}_k^{(2)} - \mathbf{D}_k^{(1)} \bigotimes \mathbf{D}_k^{(2)}) \end{array} \quad (1)$$

where

$$\mathbf{D}_i^{(j)} = \begin{cases} diag(\mathbf{Q}_0^{(j)}\mathbf{e}^T) & \text{if } i = 0 \\ \\ diag(\mathbf{P}_i^{(j)}\mathbf{e}^T) & \text{if } i > 0\,. \end{cases} \quad (2)$$

The reward and initial distribution vectors are given by

$$\mathbf{r} = \mathbf{r}^{(1)} \odot \mathbf{r}^{(2)} \text{ and } \mathbf{p}_0 = \mathbf{p}_0^{(1)} \otimes \mathbf{p}_0^{(2)} \quad (3)$$

where $\odot$ is a function that computes $\mathbf{r}(x \cdot n^{(2)} + y) = f(\mathbf{r}^{(1)}(x), \mathbf{r}^{(2)}(y))$ for some nondecreasing function $f()$ (i.e., $a \geq c \wedge b \geq d \Rightarrow f(a,b) \geq f(c,d)$).

Consider the two components of the failure and repair part in the running example (see Fig. 3), with $N = 1$. Each component has two states. $\mathbf{Q}_i = \begin{pmatrix} 0 & \lambda_i(L) \\ \mu_i & 0 \end{pmatrix}$ and $\mathbf{P}_i = \begin{pmatrix} p(1) & 0 \\ p(0) & 0 \end{pmatrix}$, with shared repair rate μ_r. The transition matrix for the composed model is $\mathbf{Q} = \begin{pmatrix} -q_{11} & \lambda_2(L) & \lambda_1(L) & 0 \\ \mu_2 + p(0)p(1)\mu_r & -q_{22} & 0 & \lambda_1(L) \\ \mu_1 + p(0)p(1)\mu_r & 0 & -q_{33} & \lambda_2(L) \\ p(0)p(0)\mu_r & \mu_1 & \mu_2 & -q_{44} \end{pmatrix}$, where q_{ii} ensures that $\mathbf{Q}$ has zero row-sums.

The descriptor matrix $\mathbf{Q}$ is defined on state space $\mathcal{S}^{(1)} \times \mathcal{S}^{(2)}$, which might be a superset of the set of reachable states. However, this aspect is not relevant for the results presented in this paper. For a practical realization of analysis algorithms, the established approaches for characterizing the subset of reachable states can be applied (see [5, 6]).

Define $\mathcal{K}^{(i)} \subseteq \mathcal{K}$ as the set containing all $k \in \mathcal{K}$ such that $\mathbf{P}_k^{(i)}$ is a stochastic matrix.

On a synchronized transition, both components change state at the same time. Nevertheless, we can consider two cases, namely the synchronous and the asynchronous case. We consider a synchronized transition to be synchronous, if it requires both components to be in particular states or if there is a rate that depends on the state of both components. If one component never needs to wait for the other component to be in a particular state, we consider the synchronized transition as asynchronous. We denote a composition as asynchronous, if all synchronized transitions are asynchronous, otherwise the composition is synchronous. This informal description is formally defined in the following definition.

Definition 3.2 *A composition is asynchronous, iff* $\mathcal{K}^{(1)} \cup \mathcal{K}^{(2)} = \mathcal{K}$*; otherwise it is synchronous. An asynchronous composition is acyclic iff* $\mathcal{K}^{(i)} = \mathcal{K}$ *for* $i \in \{1, 2\}$*. An asynchronous composition is proper if* $\mathcal{K}^{(1)} \cap \mathcal{K}^{(2)} = \emptyset$*.*

For each synchronized transition of a proper asynchronous composition, there exists one component that determines the transition rate. The other component can only accept the transition. For asynchronous composition we use a slightly different notation to indicate the difference between components according to synchronized transitions. Let $\mathbf{U}_k^{(i)} = \mathbf{P}_k^{(i)}$ for $k \in \mathcal{K}^{(i)}$ and $\mathbf{Q}_k^{(i)} = \lambda_k \mathbf{P}_k^{(i)}$ otherwise and let $\hat{\mathbf{Q}}^{(i)} = \mathbf{Q}_0^{(i)} - \mathbf{D}_0^{(i)} - \sum_{k\in\mathcal{K}^{(i)}}^{K} \mathbf{D}_k^{(i)}$. Then $\mathbf{Q}$ can be rewritten as

$$\mathbf{Q} = \begin{array}{c} \hat{\mathbf{Q}}^{(1)} \bigoplus \hat{\mathbf{Q}}^{(2)} + \sum_{k\in\mathcal{K}^{(1)}}^{K} \mathbf{Q}_k^{(1)} \bigotimes \mathbf{U}_k^{(2)} \\ + \sum_{k\in\mathcal{K}^{(2)}}^{K} \mathbf{U}_k^{(1)} \bigotimes \mathbf{Q}_k^{(2)}\,. \end{array} \quad (4)$$

For an acyclic composition, one of the sums is empty. We assume here without loss of generality that the second sum is empty in this case. We also define the matrix $\bar{\mathbf{Q}}^{(i)} = \hat{\mathbf{Q}}^{(i)} + \sum_{k\in\mathcal{K}^{(i)}}^{K} \mathbf{Q}_k^{(i)}$, which represents the behavior of component i with no transitions from the other component.

4 Preservation of Orderings by Composition

Reward-preserving preorders are extended to components by extending the second condition in definition 2.5 to all matrices of a component, uniformized using the same constant for all matrices in the continuous time models. The interesting question is, under which conditions is a reward-preserving preorder preserved by composition (i.e., is a congruence for the composition)? If order is preserved by composition, then a component can be replaced by some stochastically larger or smaller aggregate, and the results for the composed models will be stochastically larger or smaller than the results of the original model. It is easy to see that in general, the reward-preserving preorder is not preserved by composition. Consider a model with one component is an arrival process, and the other component is a finite queuing system with a reward measure based on completed jobs and a penalty for lost jobs. As the arrival process increases, the measure on the queuing system will increase up to a point, and then will decrease as more jobs are lost. Thus, some additional requirements are necessary that depend on the particular type of composition. We first consider acyclic compositions, then cyclic asynchronous compositions, and finally synchronous compositions. This sequence describes an increasing complexity of compositions that results in stricter conditions for the congruence property of the preorder.

4.1 Acyclic Composition

For an acyclic composition, we assume without loss of generality that $\mathcal{K}^{(2)} = \mathcal{K}$. The second component of an acyclic composition in this situation is called an "input process," since it only receives inputs from the other component, and does not send output to the other component. The running example minus the arrival process is an input process. A system with inputs, but without outputs, can be aggregated according to its transition matrix $\bar{\mathbf{Q}}^{(i)}$ as shown in [8]. It was shown that a system with inputs can be replaced with a smaller or larger aggregate to bound results for a given arrival process.

The first common component of an acyclic composition is called an *output process*, since its output transitions affect the state of the other component, but it receives no inputs from the other component. The arrival process to the queue in our running example is an example of an output process. Output processes often have no rewards, since they model only the arrivals to some other process, but can be viewed as having zero reward rate for each state. Typical examples are Markov Arrival Processes (MAPs) or Burst Markov Arrival Processes (BMAPs). However, our reward-preserving preorder in this case assures that starting in a larger state, the departure rate after an arbitrary time will be larger or equal to the departure rate at the same time after starting initially in a smaller state.

To classify the behavior of input and output components with respect to different environments, we use the following definition.

Definition 4.1 *A component i is* input-increasing *if $\mathbf{U}_k^{(i)}(x,y) > 0$ implies $y \succeq_\sim x$ for all $k \in \mathcal{K}^{(i)}$. Similarly, a component i is* input-decreasing *if $\mathbf{U}_k^{(i)}(x,y) > 0$ implies $x \succeq_\sim y$ for all $k \in \mathcal{K}^{(i)}$.*

Input-increasing and -decreasing systems show some form of a monotonicity property such that with an increasing/decreasing arrival rate, the reward grows/shrinks. In compositional analysis, such a behavior is important when an output process that supplies arrivals to an input model is replaced with larger or smaller aggregates, and bounds for the component should be computed. Without such a monotonic behavior, it would be unclear whether the reward value increased or decreased under a larger or smaller aggregate for the output process.

We now consider an acyclic system. The generator matrix for the composed system was shown in Eq. (4). The following theorem provides conditions under which bounds on the complete model exist when one component in the model is replaced with a larger or smaller aggregated version, based upon the preservation of $\succeq_\sim$ by matrix operations.

Theorem 4.1 *If the composition of two components (1) and (2) is as described above, then the following relations hold.*

1. *If component (2) is input-increasing (or input-decreasing and component (1) has no rewards) and $(a+)$ is a larger aggregate for (1), then the composition of $(a+)$ with (2) yields a system that is larger (or smaller) than the original system.*
2. *If component (2) is input-increasing (or input-decreasing and component (1) has no rewards) and $(a-)$ is a smaller aggregate for (1), then the composition of $(a-)$ with (2) yields a system that is smaller (or larger) than the original system.*
3. *If $(a+)$ is a larger aggregate for (2), then the composition of $(a+)$ with (1) yields a system that is larger than the original system.*
4. *If $(a-)$ is a smaller aggregate for (2), then the composition of $(a-)$ with (1) yields a system that is smaller than the original system.*

Proof. Parts 3 and 4 follow from results in [8], since the changes have no effect on the first component and are guaranteed to increase the second component, and the reward is a nondecreasing function of the rewards of all the components.

To prove part 1 of the theorem, we need to show that the complete model with a larger aggregate for the first component has states that are greater than the states of the complete model with unaggregated components. First, we show that if $x_1 \succeq_\sim^{(1)} x_2$ and $y_1 \succeq_\sim^{(2)} y_2$, then $(x_1, y_1) \succeq_\sim (x_2, y_2)$.

Define the preorder $\succeq_*$ such that if $x_1 \succeq_\sim^{(1)} x_2$ and $y_1 \succeq_\sim^{(2)} y_2$, then $(x_1, y_1) \succeq_* (x_2, y_2)$. We show that the preorder $\succeq_*$ is reward-preserving. Consider arbitrary states $(x_1, y_1), (x_2, y_2)$ such that $x_1 \succeq_\sim^{(1)} x_2$ and $y_1 \succeq_\sim^{(2)} y_2$. $\mathbf{r}(x_1, y_1) \geq \mathbf{r}(x_2, y_2)$, since the rewards are combined in a nondecreasing fashion.

Next, we must show that $\mathbf{P}((x_1, y_1)\bullet) \succeq_* \mathbf{P}((x_2, y_2)\bullet)$. For simplicity, call the vector $\mathbf{P}((x_1, y_1)\bullet)$, $\mathbf{a}$, the vectors $\mathbf{P}_k^{(1)}(x_1\bullet)$, $\mathbf{b}_k$, and the vectors $\mathbf{P}^{(2)}(y_1\bullet)$, $\mathbf{U}_k(x\bullet)$, $\mathbf{c}_k$. By assumption, there are decompositions ϕ_{xk} and ϕ_{yk} such that for all $w_1 \in \mathcal{S}^{(1)}$: $\sum_{z \succeq w_1} \mathbf{b}_{kw_1}^{(1)}(z) \geq \mathbf{P}_k^{(1)}(x_2, w_1)$,[1] and $w_2 \in \mathcal{S}^{(2)}$: $\sum_{z \succeq w_2} \mathbf{c}_{kw_2}^{(2)}(z) \geq \mathbf{P}_k^{(2)}(y_1, w_2)$. We construct a decomposition ϕ of $\mathbf{P}(x_1, y_1)$ from ϕ_{xk} and ϕ_{yk}, and need to show that for all $(w_1, w_2) \in \mathcal{S}$: $\sum_{(z_1,z_2) \succeq_* (w_1,w_2)} \mathbf{a}_{w_1,w_2}(z_1, z_2) \geq \mathbf{P}((x_2, y_2), (w_1, w_2))$.

$$\mathbf{a}_{w_1,w_2} = \mathbf{b}_{0w_1} \otimes \mathbf{e}_{y_1} + \mathbf{e}_{x_1} \otimes \mathbf{c}_{0w_2} + \sum_{k>0} \mathbf{b}_{kw_1} \otimes \mathbf{c}_{kw_2}$$

[1] The decomposition is computed and tested for $\mathbf{P}^{(1)} = \sum \mathbf{P}_i^{(1)}$ instead of $\mathbf{P}_0^{(1)}$.

We consider three cases. The first case is $w = x_2$, indicating a local transition in the second component. In that case, the first and last terms in the decomposition will be zero. The remaining term is based on the decomposition of the local transitions for the second module, which was reward-preserving. That combined with the fact that the first component does not change states, implies that $(w_1, w_2) \in \mathcal{S} : \sum_{(z_1,z_2)\succeq_*(w_1,w_2)} \mathbf{a}_{w_1,w_2}(z_1, z_2) \geq \mathbf{P}((x_2, y_2), (w_1, w_2))$.

The second case is $w \neq x_2, z \neq y_2$, indicating an output event from the first component. In that case, the first and second term of the decomposition will be zero. The remaining term is based on the decompositions of the output transitions of the first model, and the decompositions of the input events of the second. Since both sets of decompositions are reward-preserving decompositions, $(w_1, w_2) \in \mathcal{S} : \sum_{(z_1,z_2)\succeq_*(w_1,w_2)} \mathbf{a}_{w_1,w_2}(z_1, z_2) \geq \mathbf{P}((x_2, y_2), (w_1, w_2))$.

The third case is $z = y_2$, indicating a local event in the first component. Therefore, the second component in the decomposition will be zero. However, the third component need not be zero. That is, the decomposition may use an output event. However, the summation of all the decompositions is reward-preserving on the first component, and the second model is output-increasing, so any output event will leave it in a $\succeq^{(2)}_{\sim}$ state. Therefore, $(w_1, w_2) \in \mathcal{S} : \sum_{(z_1,z_2)\succeq_*(w_1,w_2)} \mathbf{a}_{w_1,w_2}(z_1, z_2) \geq \mathbf{P}((x_2, y_2), (w_1, w_2))$, $\mathbf{P}((x_1, y_1)\bullet) \succeq_* \mathbf{P}((x_2, y_2)\bullet)$, $\succeq^*$ is a reward-preserving preorder, and is therefore a subset of $\succeq_\sim$.

Given that if $x_1 \succeq^{(1)}_{\sim} x_2$ and $y_1 \succeq^{(2)}_{\sim} y_2$, then $(x_1, y_1) \succeq_\sim (x_2, y_2)$, it becomes trivial to prove part 1 of the theorem. State (x, y) in the unaggregated complete model will be mapped to state $(map(x), y)$ when the first component is aggregated. But $map(x) \succeq^{(1)}_{\sim} x$, and $y \succeq^{(2)}_{\sim} y$, so therefore $(map(x), y) \succeq_\sim (x, y)$ for all states x, y.

If component (2) is input-decreasing, the analysis is similar, except that more arrivals lead to lesser states in the second component. Therefore, a larger aggregate for component (1) will decrease the reward of the second component. If there are no rewards on the first model, then the composed system will also be smaller. The input-decreasing case is not the dual of the input-increasing case because of the reward structure of the model. Rewards may be defined on the first model, and a larger aggregate will result in a large reward on the first component, but a smaller reward on the second component. The non-decreasing combination of those rewards may be larger or smaller.

The second part of the theorem is just the reverse of the first part, and can be proved with a similar argument. □

Of course, the different ways to build bounding aggregates can be combined. For example, the input process could be replaced with some smaller or larger aggregate according to part 3 or 4 of Theorem 4.1, and afterwards a larger or smaller aggregate for component (2) could be used to build a second aggregated model that bounds the results of the first one.

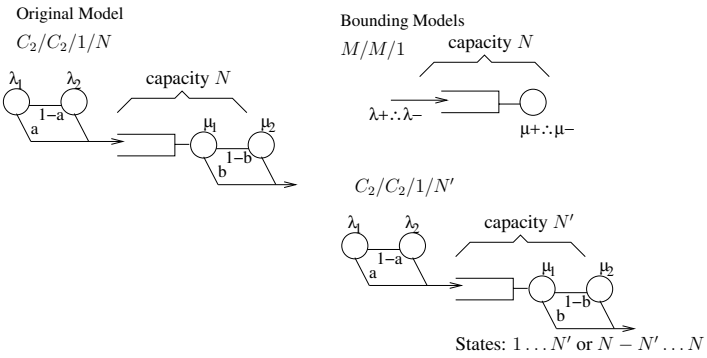

Figure 2. $C_2/C_2/1/N$ model and possible aggregates.

If we momentarily focus just on the arrival process and server in the running example, we note that the service process is an input-increasing input process with regards to the measure queue length, since arrivals immediately increase the queue length. This simplified model is shown in Figure 2. We can aggregate the input process by bounding the number of customers in the queue, or using an exponential service rate greater than either phase in the Cox-2 service process to create a lower bound on the number of customers in the queue. Similarly, we could create upper bounds on the number of customers in the queue. Additionally, we can create upper and lower bounds on the arrival process by aggregating the two state Cox-2 arrival process with a bounding, 1-state Poisson arrival process. These bounds are demonstrated in Figure 2.

4.2 Cyclic Asynchronous Composition

A cyclic composition can be created from two models that are both simultaneously input and output models. We can create a chain of such models by connecting inputs and outputs appropriately. The cyclic case is merely a simple extension of the acyclic case, and all models may be compared pairwise to determine the net effect. We analyze the three possible combinations of two components that are input-increasing or input-decreasing. Therefore, consider two input and output models A and B, such that the output of A goes to B, and the output of B goes to A.

If A and B are both input-increasing, replacing A with a larger aggregate will lead to a larger global model, as A will experience greater states locally and will have increased departures to B. The increase in arrivals to B will result in greater states in B, and the greater states in B will increase

the departures back to A, further improving the state of A. The models will behave similarly if a smaller aggregate is used.

If one component (A) is input-increasing and the other (B) is input-decreasing, no bounding results can be ensured. Replacing A with a larger aggregate will lead to greater local states and more departures. However, more departures will lead to lesser local states in B, and therefore fewer departures from B. A will receive fewer arrivals, and thus have lesser local states. This negative feedback makes it impossible for us to guarantee any bounds at this level of analysis. Similar behavior will also be observed if A is replaced with a smaller aggregate, or if B is replaced with a larger or smaller aggregate.

If both models are input-decreasing, the behavior is slightly more complicated, and no global ordering results on the states can be assured. However, if both components are input-decreasing, and component A is replaced with a larger aggregate, the states of component A will be increased, leading to more arrivals to component B. The increase in arrivals to B will lead to lesser local states in B, and fewer departures to component A. However, the decrease in arrivals to A will further improve its local behavior. Therefore, component A will always be in greater states than it would have been, and component B will be in lesser states. Therefore, the behavior will lead to bounds for rewards defined on one component, but not for global rewards that are defined as a function of both components.

4.3 Synchronous Composition

We now consider the case of synchronous composition of two components, where the two components synchronize on some transitions. The failure and repair processes in the running example demonstrate synchronous composition. As shown in Figure 3, the components fail and get repaired independently, but can synchronize to do a shared repair.

In the asynchronous case, we defined the term *input-increasing* for components that go to greater states upon receiving inputs. Analogously, since each component is affected by a synchronization, we define the term *synchronization-increasing* to describe a component that increases on a synchronization (goes to a $\succeq_\sim$ state), and the term *synchronization-decreasing* to describe a component that decreases on a synchronization. Since the reward-preserving decomposition is computed with respect to each synchronization matrix, a greater state will lead to the enabling of more synchronizations than a lesser state would. In the following theorem we use the notation $\succeq_\sim^{(i)}$ for the coarsest reward-preserving preorder for component i that can be computed with the stepwise refinement algorithm proposed in [8]. Similarly, $\succeq_\sim^{12}$ denotes the coarsest reward-preserving preorder for the composed model.

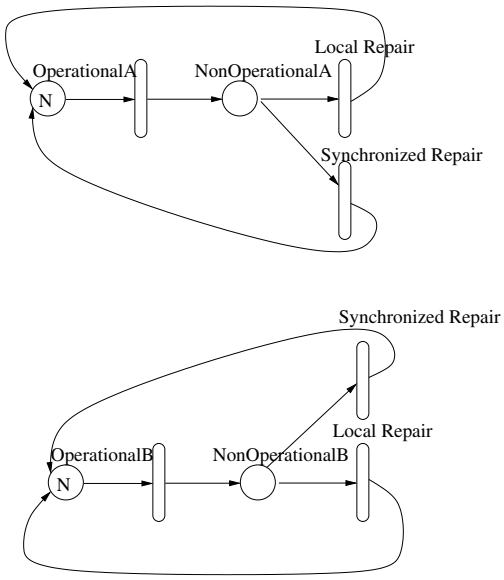

Figure 3. Model of a dependable system, with two subsystems and a synchronized, shared repair process.

Theorem 4.2 *If $\succ_\sim^{(1)}$ has been computed for component (1), $\succ_\sim^{(2)}$ has been computed for component (2), $\succeq_\sim^{(12)}$ has been computed with respect to the complete system $\mathbf{Q} = \mathbf{Q}_0^{(1)} \bigoplus \mathbf{Q}_0^{(2)} - \mathbf{D}_0^{(1)} \bigoplus \mathbf{D}_0^{(2)} + \sum_{k=1}^{K} \lambda_k (\mathbf{P}_k^{(1)} \bigotimes \mathbf{P}_k^{(2)} - \mathbf{D}_k^{(1)} \bigotimes \mathbf{D}_k^{(2)})$, and both components are synchronization-increasing, then $x_1 \succeq_\sim^{(1)} y_1 \wedge x_2 \succeq_\sim^{(2)} y_2 \Rightarrow (x_1, x_2) \succeq_\sim^{(12)} (y_1, y_2)$.*

Proof. The complete proof is omitted here and can be found in [8]. The proof is inductive using the stepwise refinement to compute $\succeq_\sim$ from some initial partition. □

Theorem 4.3 *If $\mathbf{Q} = \mathbf{Q}_0^{(1)} \bigoplus \mathbf{Q}_0^{(2)} - \mathbf{D}_0^{(1)} \bigoplus \mathbf{D}_0^{(2)} + \sum_{k=1}^{K} \lambda_k (\mathbf{P}_k^{(1)} \bigotimes \mathbf{P}_k^{(2)} - \mathbf{D}_k^{(1)} \bigotimes \mathbf{D}_k^{(2)})$ and both components are synchronization-increasing, if either component is replaced with a larger (smaller) aggregate, then the composed model will be larger (smaller).*

Proof. This follows from Theorem 4.2. If we replace component (1) with a larger aggregate, each state (x, y) of the unaggregated model will map to $(map(x), y)$ of the aggregated model. Since it is a larger aggregate, $map(x) \succeq_\sim^{(1)} x$, and it clearly follows that $y \succeq_\sim^{(2)} y$. Therefore, by Theorem 4.2, $(map(x), y) \succeq_\sim (x, y)$. □

We note that in our small example, the two components are both synchronization-increasing with respect to

the number of functional units. The state space of each component can be represented by the number of operational units, and each component has a total order on its state space with regard to the number of operational units. As more units are operational, the component is more likely to participate in a synchronization, and the synchronization leads to more operational parts (or to the same number if all are already operational).

Since the components are synchronization-increasing, we can replace either component with a larger aggregate to create a larger aggregate for the entire system. A larger aggregate can be created by combining the states $(0), (1), ..., (n)$ and replacing them with a projection of state (n) on the aggregated system. Similarly, a smaller aggregate can be created by combining the states $(n), (n+1), ..., (N)$ and replacing them with the projection of state (n) on the aggregated state space.

A special case occurs when one of the synchronizing transition matrices has fixed row sums. In that case, the component with the fixed row sum has no effect on the other component through that synchronization, since the component will always provide the same opportunity for synchronization regardless of its state. Therefore, it does not matter if the other component is synchronization-increasing or decreasing in that synchronization. This corresponds to the case of the input model in the asynchronous case. It has fixed row sums of 1 in its synchronization matrix, and has no effect on the output model.

5 Numerical Example

We introduced a running example in Section 3.3.1 to demonstrate composition and the conditions under which composition preserves bounding aggregation.

For illustrative purposes, we assign values to the model's parameters, solve it, and solve bounding aggregates for the example model. Table 1 shows the set of parameters that we use for the example model. The model has 35,280 states, and solving the model shows that it has an average queue length of 2.03. We can easily solve the example model on a personal workstation. However, since the model is for illustrative purposes, we apply a number of aggregations (as described throughout the paper) to the model in order to solve smaller, aggregated versions of the model. We construct one upper-bounding model and one lower-bounding model. In practice, the techniques could be applied to much larger, compositionally defined models that otherwise either could not be solved, or would require too much time or other resources to solve efficiently.

The upper-bounding model has exponential arrivals and departures from the front-end request-processing system, using an upper bound for the arrival rate (110) and a lower bound for the departure rate (150). Additionally, we aggregate the back-end by limiting the number of operational units for each unit type to 10, by assuming that 10 of the units are always in the failed state. Applying these aggregations results in an aggregated model with 2,541 states, or almost 14 times fewer states. Solving the aggregated model gives an average queue length of 2.19.

Parameter	**Value**
λ_1	200
λ_2	110
a	0.5
μ_1	300
μ_2	160
b	0.5
Buffer Size	20
N	20
$\lambda_1(L)$	$1.0 + 0.5 * L$
$\lambda_2(L)$	$1.0 + 0.5 * L$
Repair Rate 1	10
Repair Rate 2	10
Synchronized Repair Rate	$0.1 * OperationalA +$ $0.1 * OperationalB$

Table 1. Parameters for the Running Example

The lower-bounding model also has exponential arrivals and departures for the front-end queuing system, but uses a lower bound for the arrival rate (100) and an upper bound for the departure rate (160). Additionally, one of the two components is adjusted to ensure that it always has at least 10 operational units, and the queue buffer is limited to 10 items. Applying these aggregations to the model results in an aggregated model with 2,541 states. Solving the aggregated model gives an average queue length of 1.87.

The upper and lower bounds each have an error of less than 8% and provide strict bounds on the actual queue length. Of course the measured error depends on the parameters used in the example. Different parameters could lead to larger or smaller errors. The example model exhibits near lumpability and has low probability states that are relatively unimportant, and we expect the aggregation techniques to be particularly useful for models with these properties.

6 Conclusions

In this paper we consider the preservation of a reward-preserving preorder under composition. For different kinds of composition, it has been shown that the preorder is preserved under certain restrictions. These restrictions become more strict if the complexity of the composition increases. That is, they are stricter for cyclic than for acyclic composition and they are more strict for synchronous than for asyn-

chronous composition. Although we presented the results here for the different compositions in isolation, we also note that all three types of composition and aggregation can be used together as in the running example. The aggregation based on the asynchronous composition in the server process, as well as the synchronous composition of the component failure models, can be used to create bounding aggregates that are then composed together using the functional Kronecker composition. Each type of composition and aggregation can be applied hierarchically to create aggregates for the components, which can then composed together to create an aggregate for the complete composed model.

The preservation of the preorder allows one to use bounded aggregation for huge models, in which the components are still manageable. Of course, more experience is necessary to know for which classes of models the approach results in smaller aggregates that yield tight bounds. The major point now is the realization of efficient algorithms to compute the preorder and to generate bounding aggregates. First approaches are available in [8]. There does not seem to be many additional cases, beyond those proposed in this paper, in which composition preserves the preorder, since without the use of some sort of monotonicity property, counter-examples can be found where the pre-order is not preserved.

References

[1] P. Buchholz. On the exact and approximate analysis of hierarchical discrete time queueing networks. In H. Beilner and F. Bause, editors, *Quantitative Evaluation of Computing and Communication Systems*, volume 977 of *LNCS*, pages 150–164. Springer, 1995.

[2] P. Buchholz. Exact performance equivalence - An equivalence relation for stochastic automata. *Theoretical Computer Science*, 215(1/2):263–287, 1999.

[3] P. Buchholz. An improved method for bounding stationary measures of finite Markov processes. *Performance Evaluation*, 62(1-4):349–365, 2005.

[4] P. Buchholz. A class of hierarchical queueing networks and their analysis. *Queueing Systems*, 15(1):1994, 59-80.

[5] P. Buchholz and P. Kemper. Kronecker based matrix representations for large Markov chains. In B. Haverkort and H. H. M. Siegle, editors, *Validation of Stochastic Systems*, volume 2925 of *LNCS*, pages 256–295, 2004.

[6] G. Ciardo and A. S. Miner. A data structure for the efficient Kronecker solution of GSPNs. In P. Buchholz and M. Silva, editors, *Proc. 8th Int. Work. on Petri Nets and Performance Models*, pages 22–31, 1999.

[7] P. J. Courtois and P. Semal. Bounds for the positive eigenvectors of nonnegative matrices and for their approximation by decomposition. *Journal of the ACM*, 31(4):804–825, 1984.

[8] D. Daly. *Bounded Aggregation Techniques to Solve Large Markov Models*. PhD thesis, University of Illinois at Urbana-Champaign, 2005.

[9] D. Daly, P. Buchholz, and W. H. Sanders. A preorder relation for markov reward processes. Technical report, IBM Research Technical Report, 2005. http://domino.research.ibm.com/library/cyberdig.nsf/index.html.

[10] J. M. Fourneau and N. Pekergin. An algorithmic approach to stochastic bounds. In M. Calzarossa and S. Tucci, editors, *Performance 2002: Tutorial Lectures*, volume 2459 of *LNCS*, pages 64–88. Springer, 2002.

[11] J. Hillston. A compositional approach for performance modelling. Ph.d. thesis, University of Edinburgh, Dep. of Comp. Sc., 1994.

[12] J. Keilson and A. Kester. Monotone matrices and monotone Markov processes. *Stochastic Processes and their Applications*, 5:231–241, 1977.

[13] S. Mahevas and G. Rubino. Bound computation of dependability and performance measures. *IEEE Transactions on Computers*, 50(5):399–413, 2001.

[14] A. Müller and D. Stoyan. *Comparison Methods for Stochastic Models and Risks*. Wiley, 2002.

[15] R. R. Muntz and J. C. S. Lui. Computing bounds on steady-state availability of repairable computer systems. *Journal of the ACM*, 41(4):676–707, 1994.

[16] N. Pekergin, T. Dayar, and D. N. Alparslan. Componentwise bounds for nearly completely decomposable Markov chains using stochastic comparison and reordering. *European Journal of Operational Research*, 165:810–825, 2005.

[17] B. Plateau. On the stochastic structure of parallelism and synchronisation models for distributed algorithms. *ACM Performance Evaluation Review*, 13:142–154, 1985.

[18] B. Plateau and K. Atif. Stochastic automata networks for modeling parallel systems. *IEEE Transactions on Software Engineering*, 17(10):1093–1108, 1991.

[19] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, Princeton, NJ, 1994.

[20] M. Tremolieres, J. M. Vincent, and B. Plateau. Determination of the optimal stochastic upper bound of a Markovian generator. Rapport de Recherche RR 906-I, Institut IMAG, 1992.