Available online at www.sciencedirect.com

ScienceDirect

Theoretical Computer Science 393 (2008) 109–123

Theoretical Computer Science

www.elsevier.com/locate/tcs

Bisimulation relations for weighted automata

Peter Buchholz[*]

Informatik IV, Universität Dortmund, D-44221 Dortmund, Germany

Received 3 January 2005; received in revised form 22 October 2007; accepted 23 November 2007

Communicated by Z. Esik

Abstract

Bisimulation is a well known equivalence relation for discrete event systems and has been extended to probabilistic and stochastic systems. This paper introduces a general definition of bisimulation which can be applied to finite automata where weights and labels are assigned to transitions. It is shown that this general view contains several known bisimulations as special cases and defines naturally equivalences for different classes of models. Apart from the well known forward bisimulation, also backward bisimulation is introduced and it is shown that both types of bisimulation preserve different types of results. Furthermore it is shown that forward and backward bisimulation are congruences according to commonly known composition operations for automata.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Finite automata; Weighted automata; Equivalence; Bisimulation; Congruence; Aggregation

1. Introduction

Bisimulation as introduced by Park and Milner [37,39] is an established form of equivalence for discrete event systems like process algebras, Petri nets or automata models. Bisimulation equivalence allows one to relate equivalent systems and to reduce the state space of a system by combining equivalent states to generate an aggregated system with an equivalent behavior but with fewer states.

Originally bisimulation has been defined for systems with interleaving semantics where transitions are labeled with labels from some finite alphabet. This approach has been extended in several directions. One direction is the extension of bisimulation to systems where transitions receive a second label describing a valuation or a weight for the corresponding transition. A typical example are probabilistic systems, where transitions are additionally labeled with probabilities. Different types of probabilistic systems exist (see [4,44] for an overview). Bisimulation for probabilistic processes has been introduced by several authors, important sources are [32,35]. If apart from the functional behavior also timed behavior has to be described by a model, timing is also introduced using valuation of transitions or states of systems. A widely used approach is to assume stochastic timing as common in stochastic automata [40,11], stochastic process algebras [31,27] or stochastic Petri nets [1]. It is not surprising that bisimulation has been adopted for these

[*] Tel.: +49 231 755 4746; fax: +49 231 755 4730.
E-mail address: peter.buchholz@udo.edu.

0304-3975/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2007.11.018

types of models. Performance bisimulation, as it is sometimes denoted, has been defined for stochastic automata [9], stochastic process algebras [8,28,29], stochastic Petri nets [10] and Markov chains with transition labels [7].

Although the ideas underlying the above mentioned definitions of bisimulation are very similar, they are all presented in different settings using different notation. However, a universal definition of bisimulation which can be applied to a wide class of model types such that the different forms of bisimulation can all be seen as specific cases would help to unify system analysis. A presentation of such a general definition is the goal of this paper. We will show that bisimulation can be naturally defined for automata where each transition receives two labels, one from a finite alphabet and a second label from some semiring. This yields the well known class of weighted automata [34] which is slightly extended here by definition of some composition operations. Based on the semiring valuation of transitions, we can define bisimulation in a very general form. By considering specific semirings, this general form corresponds to the standard form of bisimulation as defined for the specific model types mentioned above. Additionally, the general form of bisimulation defines naturally bisimulation equivalence for model types where this equivalence has not been considered yet like (max, $+$) automata [3,24]. The concept of forward bisimulation for general transitions valuations has been first presented for the process algebra GPA [14]. Here we apply the approach for weighted automata and extend it slightly. Furthermore, apart from the standard bisimulation, we consider also two variants of backward or inverse bisimulation in the setting of weighted automata which has been defined for stochastic systems [11] and is also useful for untimed models as shown in [12]. The first variant of backward bisimulation follows directly from forward bisimulation exploiting the properties of the used operations on matrices, whereas the second variant seems to be new in the context of weighted automata. Like bisimulation, backward bisimulation can be defined in a general setting and applies as special case to a large number of different concrete model types. We show that all proposed bisimulations are congruence relations according to natural composition operations for weighted automata.

One goal of this paper is to present different forms of bisimulation relations using a matrix notation that can be easily applied as a base for the implementation of algorithms to compute the largest bisimulation for a given weighted automaton. According to this bisimulation an aggregated automaton with a smaller state space but an equivalent behavior can be built. Since we prove that the different bisimulation relations are congruences according to natural composition operations for weighted automata, aggregation and composition can be interleaved for compositional analysis of complex systems. This approach is well established for functional analysis [19] and more recently also for dependability or performance analysis [27,29,31] using in both cases forward bisimulation. Backward bisimulation, that is also introduced in this paper, is less established and only applied in very few approaches for stochastic modeling [41]. For other automata types, especially for max $/+$ automata or related models that are often used for the analysis of discrete event systems [3,24], compositional analysis and bisimulation are unknown yet. The paper therefore introduces a step towards a generic approach for compositional analysis of discrete event systems with finite state spaces.

The structure of the paper is as follows. In the next section we present the basic automaton model together with some composition operations. Afterwards we introduce forward bisimulation. The next section describes backward bisimulation as an alternative equivalence relation. All concepts are accompanied by simple examples to show the generality and flexibility of the proposed automaton model and the proposed equivalence relations. The paper ends with the conclusions.

2. A weighted automata model

To present our general automata model, we first introduce semirings which are needed to define labels for transitions. Afterwards weighted automata are defined.

Definition 2.1. A semiring $(\mathbb{K}, \widehat{+}, \widehat{\cdot}, \mathbb{O}, \mathbb{1})$ is a set $\mathbb{K}$ with binary operations $\widehat{+}$ and $\widehat{\cdot}$ defined on $\mathbb{K}$ such that the following axioms are satisfied:

(1) $\widehat{+}$, $\widehat{\cdot}$ are associative,
(2) $\widehat{+}$ is commutative,
(3) right and left distributive laws hold for $\widehat{+}$ and $\widehat{\cdot}$,
(4) $\mathbb{O}$ and $\mathbb{1}$ are the additive and multiplicative identities with $\mathbb{O} \neq \mathbb{1}$,
(5) for all $k \in \mathbb{K}$ $k \widehat{\cdot} \mathbb{O} = \mathbb{O} \widehat{\cdot} k = \mathbb{O}$ holds.

To make the notation simpler we use sometimes $\mathbb{K}$ for the whole semiring. Furthermore we use the usual abbreviations $\widehat{\prod}_{i=1}^{N} l_i = l_1 \hat{\cdot} l_2 \hat{\cdot} \ldots \hat{\cdot} l_N$, $ab = a \hat{\cdot} b$ and $\widehat{\sum}_{i=1}^{N} l_i = l_1 \hat{+} l_2 \hat{+} \ldots \hat{+} l_N$.

Vectors and matrices over a semiring $\mathbb{K}$ can be defined in a natural way. Let $\mathbb{K}^{n,m}$ be the set of all matrices with n rows and m columns with elements from $\mathbb{K}$. $\mathbf{I}$ is the identity matrix over semiring $\mathbb{K}$, $\mathbf{e}_x \in \mathbb{K}^{1,n}$ is a row vector with $\mathbb{1}$ in position x $(0 \le x < n)$ and $\mathbb{0}$ elsewhere and $\mathbf{e}$ is a row vector where all elements equal to $\mathbb{1}$. $\cdot^T$ is the transposed of a vector or matrix. Addition and multiplication of matrices $\mathbf{A} \in \mathbb{K}^{n,m}$ and $\mathbf{B} \in \mathbb{K}^{r,s}$ can be naturally defined as

$$\mathbf{C} = \mathbf{A} \hat{+} \mathbf{B} \quad \text{with } \mathbf{C} \in \mathbb{K}^{n,m} : \ \mathbf{C}(x,y) = \mathbf{A}(x,y) \hat{+} \mathbf{B}(x,y) \quad \text{if } n = r \text{ and } m = s,$$

$$\mathbf{D} = \mathbf{A} \hat{\cdot} \mathbf{B} \quad \text{with } \mathbf{D} \in \mathbb{K}^{n,s} : \ \mathbf{D}(x,y) = \widehat{\sum}_{z=0}^{m-1} \mathbf{A}(x,z) \hat{\cdot} \mathbf{B}(z,y) \quad \text{if } m = r.$$

Similarly, the Kronecker product of matrices which is required for the matrix representation of composed automata is defined as

$$\mathbf{F} = \mathbf{A} \hat{\otimes} \mathbf{B} \quad \text{with } \mathbf{F} \in \mathbb{K}^{nr,ms} : \ \mathbf{F} = \begin{pmatrix} \mathbf{A}(0,0) \hat{\cdot} \mathbf{B} & \cdots & \mathbf{A}(0,m-1) \hat{\cdot} \mathbf{B} \\ \vdots & \ddots & \vdots \\ \mathbf{A}(n-1,0) \hat{\cdot} \mathbf{B} & \cdots & \mathbf{A}(n-1,m-1) \hat{\cdot} \mathbf{B} \end{pmatrix}$$

and the Kronecker sum is given by

$$\mathbf{G} = \mathbf{A} \hat{\oplus} \mathbf{B} \quad \text{with } \mathbf{G} \in \mathbb{K}^{nr,ms} : \mathbf{G} = \mathbf{A} \hat{\otimes} \mathbf{I} \hat{+} \mathbf{I} \hat{\otimes} \mathbf{B} \quad \text{if } n = m \text{ and } r = s.$$

We consider weighted automata over finite alphabets and with a finite state space. For the representation we use a matrix notation following [6,23] which allows an elegant and compact formulation of the theory.

Definition 2.2. A finite weighted automaton with n states over the finite alphabet Act and semiring $\mathbb{K}$ equals $\mathcal{A} = (\mathbf{a}, (\mathbf{T}_l)_{l \in Act}, \mathbf{b})$ where $\mathbf{a} \in \mathbb{K}^{1,n}$, $\mathbf{T}_l \in \mathbb{K}^{n,n}$ and $\mathbf{b} \in \mathbb{K}^{n,1}$.

$\mathbf{a}(x)$ is the initial weight of state x and $\mathbf{b}(x)$ defines the final weight x which is used to represent additional cost of paths ending in state x (see the examples below). The interpretation of these values depends on the concrete automaton model. $\mathbf{T}_l(x,y)$ equals the weight of the transition between states x and y that is labeled with $l \in Act$. Weight $\mathbb{0}$ is usually interpreted that no such transition exists. Following [6] we denote by $[n]$ the set $\{0, \ldots, n-1\}$.

We assume that transitions of an automaton can be observed through their labels. Consequently, an action sequence is defined as a sequence $seq = l_1, \ldots, l_k$ with $l_i \in Act$. Let Act^* be the set of all finite action sequences. The weight of action sequence seq starting in state x is given by

$$c_x(seq) = \mathbf{a}(x) \hat{\cdot} \mathbf{e}_x \hat{\cdot} \left(\widehat{\prod}_{i=1}^{k} \mathbf{T}_{l_i} \right) \hat{\cdot} \mathbf{b}, \tag{1}$$

where multiplication and addition are defined over the semiring. The above computation of weights assumes that a specific initial state is known. Alternatively one can consider the case that vector $\mathbf{a}$ defines the weights of initial states. The weights of an action sequence are then defined as

$$c(seq) = \mathbf{a} \hat{\cdot} \left(\widehat{\prod}_{i=1}^{k} \mathbf{T}_{l_i} \right) \hat{\cdot} \mathbf{b}. \tag{2}$$

Apart from the weights of an action sequence we consider possible terminating states and the weights of reaching those states. These values are described by a vector

$$\mathbf{d}_{seq} = \mathbf{a} \hat{\cdot} \left(\widehat{\prod}_{i=1}^{k} \mathbf{T}_{l_i} \right). \tag{3}$$

Obviously the following relation holds:

$$c(seq) = \mathbf{d}_{seq} \hat{\cdot} \mathbf{b}. \tag{4}$$

For the composition of automata different operations are available and can in principle be extended to weighted automata. However, for weighted automata the composition of weights has to be defined in an appropriate way. Usually different possibilities exist, as one can see by comparing different forms of computing rates of joint transitions in stochastic process algebras [30]. Here we use the composition operations which have been introduced in [16] and naturally extend composition operations for stochastic and probabilistic automata models.

Definition 2.3. Let $\mathcal{A}_1 = (\mathbf{a}^{(1)}, (\mathbf{T}_l^{(1)})_{l\in Act_1}, \mathbf{b}^{(1)})$ and $\mathcal{A}_2 = (\mathbf{a}^{(2)}, (\mathbf{T}_l^{(2)})_{l\in Act_2}, \mathbf{b}^{(2)})$ be two weighted automata over the same semiring $\mathbb{K}$.

Let $\mathbf{T}_l^{(i)} = \mathbb{O}$ for $l \notin Act_i$.

(1) The direct sum of automata is defined as an automaton $\mathcal{A}_0 = \mathcal{A}_1 + \mathcal{A}_2$ with $Act_0 = Act_1 \cup Act_2$,

$$\mathbf{a}^{(0)} = \left(\mathbf{a}^{(1)}, \mathbf{a}^{(2)}\right),$$
$$\mathbf{b}^{(0)} = \begin{pmatrix}\mathbf{b}^{(1)}\\ \mathbf{b}^{(2)}\end{pmatrix}, \qquad \mathbf{T}_l^{(0)} = \begin{pmatrix}\mathbf{T}_l^{(1)} & \mathbb{O}\\ \mathbb{O} & \mathbf{T}_l^{(2)}\end{pmatrix} \quad \text{for all } l \in Act_0.$$

(2) The direct product of automata is defined as an automaton $\mathcal{A}_0 = \mathcal{A}_1 \cdot \mathcal{A}_2$ with $Act_0 = Act_1 \cup Act_2 \cup \{\tau\}$,

$$\mathbf{a}^{(0)} = \left(\mathbf{a}^{(1)}, \mathbb{O}\right), \qquad \mathbf{b}^{(0)} = \begin{pmatrix}\mathbb{O}\\ \mathbf{b}^{(2)}\end{pmatrix}, \qquad \mathbf{T}_l^{(0)} = \begin{pmatrix}\mathbf{T}_l^{(1)} & \mathbb{O}\\ \mathbb{O} & \mathbf{T}_l^{(2)}\end{pmatrix} \text{ if } l \in Act_0 \setminus \{\tau\}$$
$$\mathbf{T}_\tau^{(0)} = \begin{pmatrix}\mathbf{T}_\tau^{(1)} & \mathbf{b}^{(1)}\mathbf{a}^{(2)}\\ \mathbb{O} & \mathbf{T}_\tau^{(2)}\end{pmatrix} \text{ if } l = \tau.$$

Label τ is used for transitions that start in a state from automaton 1 and end in a state from automaton 2.

(3) The synchronized product over a set $Act_c \subseteq Act_1 \cap Act_2$ is defined as an automaton $\mathcal{A}_0 = \mathcal{A}_1 \|_{Act_c} \mathcal{A}_2$ with $Act_0 = Act_1 \cup Act_2$,

$$\mathbf{a}^{(0)} = \mathbf{a}^{(1)} \widehat{\otimes}\, \mathbf{a}^{(2)}, \qquad \mathbf{b}^{(0)} = \mathbf{b}^{(1)} \widehat{\otimes}\, \mathbf{b}^{(2)},$$
$$\mathbf{T}_l^{(0)} = \mathbf{T}_l^{(1)} \widehat{\otimes}\, \mathbf{T}_l^{(2)} \text{ if } l \in Act_c \quad \mathbf{T}_l^{(0)} = \mathbf{T}_l^{(1)} \widehat{\oplus}\, \mathbf{T}_l^{(2)} \text{ if } l \in Act_0 \setminus Act_c.$$

The free product results from a synchronous product with $Act_c = \emptyset$.

The direct sum and the direct product have been proposed for weighted automata and also the matrix representation is known [6]. The synchronized and free product have been defined similarly for the process algebra GPA [14] and in several concrete automata models like untimed [15], stochastic [40] or max/+ automata [24]. Composition approaches can be used to build complex automata from simpler models and are therefore very important for structured system specifications and analysis.

Examples. The definition includes well known automata models as specific cases. Different automata types arise if we choose different semirings for the transition weights.

By choosing the semiring $(\mathbb{B}, \vee, \wedge, 0, 1)$, we obtain finite automata with labeled transitions. $\mathbf{T}_l(x, y) = 1$ indicates that an l-labeled transition between x and y exists, $\mathbf{T}_l(x, y) = 0$ shows that no such transition exists. Vector $\mathbf{a}$ contains a 1 in position x, if x is an/the initial state. In the usual case with one initial state x, $\mathbf{a}$ equals $\mathbf{e}_x$. Vector $\mathbf{b}$ is used to indicate terminating states. A 1 is assigned to terminating states and a 0 to the remaining states. Via (2), the weights of a sequence *seq* can be computed. All action sequences have weights 0 or 1. Weight 1 indicates that the action sequence can be realized, i.e., an action sequence starting in an initial state, labeled with *seq* and ending in a terminating state exists. Weight 0 indicates that no such sequence exists. In a similar way $\mathbf{d}_{seq}(x) = 1$ indicates that the automaton can be in state x after starting in an initial state and observing *seq*.

If we choose $(\mathbb{R}_+, +, \cdot, 0, 1)$ and additionally assume that $\sum_{x=0}^{n-1} \mathbf{a}(x) = 1$ and $\sum_{y=0}^{n-1} \mathbf{T}_l(x, y) = 0$ or 1 for all $l \in Act$ and all $x \in \mathcal{S}$, we obtain the model of probabilistic processes as defined in [35] that is denoted as the reactive model [44]. In this model, vector $\mathbf{b}$ is not required and we may use $\mathbf{b} = \mathbf{e}$ because 1 is the neutral element of multiplication. The weights of an action sequence *seq* in this model are either 0 or 1. As for the basic model, a 1 indicates that an action sequence is possible, 0 indicates an impossible action sequence. $\mathbf{d}_{seq}(x)$ contains the probability of being in state x after starting in state y with probability $\mathbf{a}(y)$ and observing *seq*.

A slightly different model of probabilistic automata can be defined on the same semiring used in the previous example, but for the matrices $\sum_{l\in Act} \sum_{y=0}^{n-1} \mathbf{T}_l(x, y) = 1$ or 0 is assumed. This model is sometimes denoted as the generative probabilistic model [4,44]. States without outgoing transitions indicate terminal states. $c(seq)$ describes the probability of observing *seq* when the automaton starts with probability $\mathbf{a}(x)$ in state x. Probability $c(seq)$ is from the interval [0, 1].

A third form of probabilistic automata is defined using the semiring $([0, 1], \max, \cdot, 0, 1)$. $c(seq)$ describes in this model the weight of the action sequence with the highest probability that is labeled with *seq*. Similarly $\mathbf{d}_{seq}(x)$ describes the weight of the action sequence with the highest probability labeled with *seq* and ending in state x.

To define stochastic automata or process algebras [40,11,31,27] the semiring $(\mathbb{R}_+, +, \cdot, 0, 1)$ is used. As in the probabilistic case $\sum_{x=0}^{n-1} \mathbf{a}(x) = 1$ is assumed because vector $\mathbf{a}$ includes the initial probabilities. For the elements of the matrices $\mathbf{T}_l$ no additional restrictions are required. The interpretation is now that the system behaves like a continuous time Markov chain with labeled transitions. The behavior is slightly more complex than in the previous cases. We only present a very brief overview, for details and methods to compute additional results we refer to the literature (e.g., [43]). Let $\mathbf{p}$ be a probability vector, then the probability of observing $l \in Act$ as the next transition label is given by

$$prob(\mathbf{p}, l) = \begin{cases} \mathbf{p} \cdot \mathbf{T}_l \cdot \mathbf{e}^T \Big/ \left(\sum_{l \in Act} \mathbf{p} \cdot \mathbf{T}_l \cdot \mathbf{e}^T \right) & \text{if } \mathbf{p} \cdot \mathbf{T}_l \cdot \mathbf{e}^T > 0 \\ 0 & \text{otherwise} \end{cases}$$

$prob(\mathbf{p}, l)$ defines a conditional probability distribution over $l \in Act$. The condition is defined by $\mathbf{p}$, the current distribution over the set of states. If $prob(\mathbf{p}, l) > 0$, then the conditional distribution of the automata states after observing l is given by

$$\mathbf{p} \cdot \mathbf{T}_l / (\mathbf{p} \cdot \mathbf{T}_l \cdot \mathbf{e}^T).$$

With these equations one can compute the probability observing action sequence *seq* starting with distribution $\mathbf{p}$. Stochastic automata describe apart from the probabilistic behavior also timed behavior. Starting with probability distribution $\mathbf{p}$, the time before the next transition occurs is exponentially distributed with rate $\mathbf{p} \cdot (\sum_{l \in Act} \mathbf{T}_l \cdot \mathbf{e}^T)$.

Other typical examples for automata over semirings are linear systems from min/plus or max/plus algebra as often used to analyze discrete event systems [3]. Consider the semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ and interpret the values as transition cost. Then $c(seq)$ describes the minimal cost of an action sequence labeled with *seq*. Cost of an action sequence result in this model from the sum of cost for each step. Over the semiring $(\mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$ $c(seq)$ describes the maximal cost over an action sequence labeled with *seq*. As before cost result from the sum of cost of each step. Over $(\mathbb{R}_+ \cup \{\infty\} \cup -\infty, \max, \min, -\infty, \infty)$ where $\mathbf{a} = \infty \cdot \mathbf{e}$, $c(seq)$ describes the maximal cost of a single transition if the transition with the minimal cost to reach the state is always chosen. In a similar way, over $(\mathbb{R}_+ \cup \{\infty\} \cup -\infty, \min, \max, \infty, -\infty)$ with $\mathbf{a} = \mathbf{b} = -\infty \cdot \mathbf{e}$, $c(seq)$ describes the minimal cost of a transition sequence if always the transition with the maximum cost is chosen. In all cases the interpretation of $c(seq)$ and $\mathbf{d}_{seq}$ is straightforward.

As a last example we consider the semiring $(\mathbb{R}_+^{k \times k}, +, \cdot, \mathbf{0}, \mathbf{I})$, where $\mathbf{I}$ is the $k \times k$ identity matrix and $\mathbf{0}$ is a $k \times k$ matrix with all elements equal to 0. For this example each state of the automaton represents an automaton with k states. Therefore the model may be used for hierarchical automata descriptions as used in the stochastic case in [12].

Composition operations can be applied in all mentioned cases, but not in all cases the resulting automata observe the specific conditions of the composed automata. In particular for probabilistic automata, where the sum of outgoing transitions according to all or one label is either 0 or 1, the condition is not necessarily preserved after composition using direct, free or synchronized products.

3. Forward bisimulation for weighted automata

The use of bisimulation has a long history and various forms exist for different systems. The basic form of bisimulation considers discrete state spaces and often also finite branching is assumed [35] (i.e., each state has only a finite number of successors). If bisimulation is applied for practical system analysis, then often even finite state spaces are assumed [15,25,42]. This is exactly the setting we use as our starting point, namely automata with finite state spaces, and for these models we present a generalization of bisimulation. Additionally, bisimulation has been recently generalized to systems with continuous state spaces [22,20]. This class of models is, of course, not captured by the approach presented here, but a similar generalization should be possible.

The notion of bisimulation for weighted automata has been first proposed in [14] in a process algebraic setting and in [16] a weak version of bisimulation has been introduced for weighted automata over idempotent semirings. Here we extend the basic notion of bisimulation by characterizing it as a fixed point of partition refinement and by applying it for weighted automata which slightly differ from the process algebra used in [14].

Before bisimulation is introduced, we define the concept of relational simulation which is very useful for proving results based on the matrix representation of weighted automata. For a relation $\mathcal{R}$ between two sets $[n]$ and $[m]$ we define a relational matrix $\mathbf{V} \in \mathbb{K}^{n,m}$ with $\mathbf{V}(x, y) = \mathbb{1}$ if $x \mathcal{R} y$ and $\mathbb{0}$ otherwise. We identify relations with the corresponding relational matrix. The following definition and the subsequent theorems can be found in [6, Chap. 9]. We assume in the sequel that related automata are defined over the same semiring.

Definition 3.1. Let $\mathcal{A}_1 = (\mathbf{a}^{(1)}, (\mathbf{T}_l^{(1)})_{l\in Act}, \mathbf{b}^{(1)})$ and $\mathcal{A}_2 = (\mathbf{a}^{(2)}, (\mathbf{T}_l^{(2)})_{l\in Act}, \mathbf{b}^{(2)})$ be two weighted automata and $\mathbf{V}$ a relational matrix such that

$$\mathbf{a}^{(1)}\mathbf{V} = \mathbf{a}^{(2)}, \qquad \mathbf{T}_l^{(1)}\mathbf{V} = \mathbf{V}\mathbf{T}_l^{(2)} \quad \text{for all } l \in Act \text{ and } \mathbf{b}^{(1)} = \mathbf{V}\mathbf{b}^{(2)}$$

then $\mathbf{V}$ is a relational simulation $\mathcal{A}_1 \longrightarrow \mathcal{A}_2$.

If $\mathbf{V}$ is additionally a (surjective) function, then it is a (surjective) functional simulation.

Theorem 3.1. *If* $\mathbf{V}$ *is a relational simulation* $\mathcal{A}_1 \longrightarrow \mathcal{A}_2$, *then* $c^{(1)}(seq) = c^{(2)}(seq)$ *for all* $seq \in Act^*$.

Proof. We first show $\mathbf{d}_{seq}^{(1)}\mathbf{V} = \mathbf{d}_{seq}^{(2)}$.

$$\begin{aligned}\mathbf{d}_{seq}^{(1)}\mathbf{V} &= \mathbf{a}^{(1)}\hat{\prod}_{i=1}^{k}\mathbf{T}_{l_i}^{(1)}\mathbf{b}^{(1)}\mathbf{V} &&= \mathbf{a}^{(1)}\hat{\prod}_{i=1}^{k}\mathbf{T}_{l_i}^{(1)}\mathbf{V}\mathbf{b}^{(2)} \\ &= \dots &&= \mathbf{a}^{(2)}\hat{\prod}_{i=1}^{k}\mathbf{T}_{l_i}^{(2)}\mathbf{b}^{(2)} \qquad = \mathbf{d}_{seq}^{(2)}.\end{aligned}$$

This implies

$$c^{(1)}(seq) = \mathbf{d}_{seq}^{(1)}\mathbf{b}^{(1)} = \mathbf{d}_{seq}^{(1)}\mathbf{V}\mathbf{b}^{(2)} = \mathbf{d}_{seq}^{(2)}\mathbf{b}^{(2)} = c^{(2)}(seq). \quad \square$$

The following theorem shows that the product of relational simulations is a relational simulation.

Theorem 3.2. *Let* $\mathbf{V}_1$ *be a relational simulation* $\mathcal{A}_1 \longrightarrow \mathcal{A}_2$ *and* $\mathbf{V}_2$ *be a relational simulation* $\mathcal{A}_2 \longrightarrow \mathcal{A}_3$, *then* $\mathbf{V}_1\mathbf{V}_2$ *is a relational simulation* $\mathcal{A}_1 \longrightarrow \mathcal{A}_3$.

Proof. The proof is straightforward since

$$\mathbf{b}^{(1)} = \mathbf{V}_1\mathbf{b}^{(2)} = \mathbf{V}_1\mathbf{V}_2\mathbf{b}^{(3)}, \mathbf{T}_l^{(1)}\mathbf{V}^{(1)}\mathbf{V}^{(2)} = \mathbf{V}_1\mathbf{T}_l^{(2)}\mathbf{V}^{(2)} = \mathbf{V}_1\mathbf{V}_2\mathbf{T}_l^{(3)}$$

for all $l \in Act$ and $\mathbf{a}^{(3)}\mathbf{V}^{(1)}\mathbf{V}^{(2)} = \mathbf{a}^{(2)}\mathbf{V}_2 = \mathbf{a}^{(3)}$. $\square$

For system analysis the concept of an aggregated automaton which is an automaton with a smaller state space is very useful. This in particular holds if the aggregated automaton shows a similar or identical behavior than the original automaton.

Definition 3.2. For a surjective function $\mathbf{V}$ from $[n]$ to $[m]$ we define a distributor matrix $\mathbf{W} \in \mathbb{K}^{m,n}$ as a matrix with row sum $\mathbb{1}$ (i.e., $\mathbf{W}\mathbf{e}^T = \mathbf{e}^T$) according to some appropriate weight vector $\mathbf{w} \in \mathbb{K}^{1,n}$ as $\mathbf{W} = (diag(\mathbf{w})\mathbf{V})^T$ where $diag(\mathbf{w})$ is a $n \times n$ diagonal matrix with $\mathbf{w}(i)$ in position (i, i).

It is easy to show that $\mathbf{W}\hat{\cdot}\mathbf{V} = \mathbf{I}$ holds by construction of matrix $\mathbf{W}$.

Definition 3.3. For a weighted automaton $\mathcal{A} = (\mathbf{a}, \mathbf{T} = (\mathbf{T}_l)_{l\in Act}, \mathbf{b})$ with n states, a surjective function $\mathbf{V}$ from $[n]$ to $[m]$ and a distributor matrix $\mathbf{W}$ the corresponding aggregated automaton $\widetilde{\mathcal{A}} = (\tilde{\mathbf{a}}, (\tilde{\mathbf{T}}_l)_{l\in Act}, \tilde{\mathbf{b}})$ with m states is defined as

$$\tilde{\mathbf{a}} = \mathbf{a}\mathbf{V}, \qquad \tilde{\mathbf{T}}_l = \mathbf{W}\mathbf{T}_l\mathbf{V} \quad \text{for all } l \in Act \text{ and } \tilde{\mathbf{b}} = \mathbf{W}\mathbf{b}. \tag{5}$$

The aggregation is transition value preserving if for all $z \in [m]$ and for all $x \in [n]$ with $\mathbf{V}(x, z) = \mathbb{1}$, $\mathbf{e}_x\mathbf{T}_l\mathbf{e}^T = \mathbf{e}_z\tilde{\mathbf{T}}_l\mathbf{e}^T$ for all $l \in Act$.

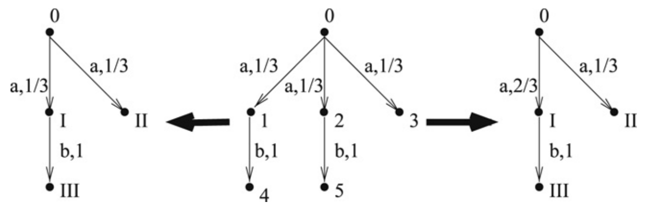

Fig. 1. A probabilistic automaton $\mathcal{A}$ and aggregated automata $\widetilde{\mathcal{A}}^*$ according to different semirings.

An aggregated automaton $\widetilde{\mathcal{A}}$ is behaviorally equivalent to the entire automaton $\mathcal{A}$ from which it is originated, if $c(seq) = \tilde{c}(seq)$ for all sequences seq where $\tilde{c}(seq)$ is computed for $\widetilde{\mathcal{A}}$ using (2). The concept of aggregation of states according to some weight vector is well known in stochastic modeling [9,18] but has, to the best of our knowledge, not really applied for other automata models. Observe that a surjective function $\mathbf{V}$ from $[n]$ to $[m]$ induces an equivalence relation on $[n]$ by relating $x, y \in [n]$ if $\mathbf{V}(x, z) = \mathbf{V}(y, z)$ for all $z \in [m]$. Since $\mathbf{V}$ is a surjective function this is equivalent to $\mathbf{V}(x, z) = \mathbf{V}(y, z) = \mathbb{1}$ for some z. Thus, $\mathbf{V}$ is the quotient map of the equivalence relation $\mathcal{E}$. We sometimes denote this equivalence relation as bisimulation instead of speaking of the quotient map of the equivalence relation. Furthermore, $\mathcal{E}_z \subseteq [n]$ $(z \in [m])$ is an equivalence class of the relation.

Definition 3.4. If $\mathbf{V}$ is a surjective functional simulation $\mathcal{A} \longrightarrow \widetilde{\mathcal{A}}$ where $\widetilde{\mathcal{A}}$ is the aggregated automaton as defined above, then $\mathbf{V}$ is a bisimulation.

For automata defined on the semiring $(\mathbb{B}, \vee, \wedge, 0, 1)$, the definition coincides with the standard notion of bisimulation [39,37] applied to finite state automata. The following corollary follows from the definition of surjective functional simulations.

Corollary 3.3. *For an automaton $\mathcal{A}$ and the aggregated automaton $\widetilde{\mathcal{A}}$ according to some bisimulation $\mathbf{V}$ and all $z \in [m]$*

$$\tilde{\mathbf{b}}(z) = \mathbf{b}(x) \quad \text{and} \quad \widehat{\sum}_{y \in \mathcal{E}_w} \mathbf{T}_l(x, y) = \tilde{\mathbf{T}}_l(z, w)$$

holds for all $x \in \mathcal{E}_z$ and all $w \in [m]$.

The corollary implies that aggregated automata that are built according to some bisimulation are invariant under the choice of different weight vectors and that the aggregation is transition value preserving.

Since a bisimulation is a relational simulation the following corollary follows from Theorem 3.1.

Corollary 3.4. *Let $\widetilde{\mathcal{A}}$ be an aggregated automaton resulting from automaton $\mathcal{A}$ according to some bisimulation $\mathbf{V}$, then*

(1) *$\mathcal{A}$ and $\widetilde{\mathcal{A}}$ are behaviorally equivalent and*
(2) *$\tilde{\mathbf{d}}_{seq}(z) = \widehat{\sum}_{x \in \mathcal{E}_z} \mathbf{d}_{seq}(x)$ for all $z \in [m]$.*

Before we proceed examples are introduced to clarify the concept for some specific classes of automata.

Examples. Fig. 1 shows in the middle a probabilistic automaton with 6 states and $Act = \{a, b\}$. Let additionally be $\mathbf{a} = (1, 0, 0, 0, 0, 0)$ and $\mathbf{b} = (0, 0, 0, 1, 1, 1)$. If the automaton is built over the semiring $(\mathbb{R}_+, +, \cdot, 0, 1)$, then it is of the type defined in [35]. The corresponding aggregated automaton according to bisimulation relation $\mathcal{R}$ defining matrix

$$\mathbf{V} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}^T$$

is shown on the right side. The aggregated automaton is independent of the weight vector $\mathbf{w}$ as long as the resulting distributor matrix $\mathbf{W}$ observes the conditions of Definition 3.2. The vectors for $\widetilde{\mathcal{A}}$ are $\tilde{\mathbf{a}} = (1, 0, 0, 0)$ and $\tilde{\mathbf{b}} = (0, 0, 1, 1)$. This bisimulation is the largest bisimulation for this automaton.

If we interpret the previous probabilistic automaton over the semiring $([0, 1], \max, \cdot, 0, 1)$, relation $\mathcal{R}$ is still the largest bisimulation, but the aggregated automaton differs from the previous one, because maximization instead of addition is used to compute transition valuation of the aggregated automaton. The resulting aggregated automaton is shown on the left side of Fig. 1. The corresponding vectors are identical to the vectors of the previous aggregation.

As another example we consider an automaton over the semiring $(\mathbb{R}_+ \cup \infty, \min, +, \infty, 0)$. The automaton has 6 states all transitions are labeled with a single label l. Transition values are given in the following matrix.

$$\mathbf{T}_l = \begin{pmatrix} 1 & 2 & \infty & 2 & \infty & \infty \\ \infty & \infty & 1 & \infty & 2 & \infty \\ 1 & \infty & \infty & 2 & 1 & \infty \\ \infty & 2 & 1 & 2 & 1 & 1 \\ 2 & 1 & \infty & 1 & \infty & 1 \\ \infty & 2 & 1 & \infty & 1 & \infty \end{pmatrix}. \tag{6}$$

Vector $\mathbf{a}$ equals $(0, \infty, \infty, \infty, \infty, \infty)$ and $\mathbf{b}$ is given by $(\infty, \infty, \infty, 3, 3, 1)$. Possible transition sequences are all of the form l^k, where $k \in \mathbb{N}$ describes the length of the sequence. For different sequences we obtain the following results.

seq	l^0	l^1	l^2	l^3	l^4
$\mathbf{d}_{seq}$	$\begin{pmatrix} 0 \\ \infty \\ \infty \\ \infty \\ \infty \\ \infty \end{pmatrix}$	$\begin{pmatrix} 1 \\ 2 \\ \infty \\ 2 \\ \infty \\ \infty \end{pmatrix}$	$\begin{pmatrix} 2 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{pmatrix}$	$\begin{pmatrix} 3 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \end{pmatrix}$	$\begin{pmatrix} 4 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \end{pmatrix}$.
$c(seq)$	∞	5	4	5	6

The largest bisimulation for the automaton is given by

$$\mathbf{V} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^T$$

and yields the aggregated automaton defined by matrix

$$\tilde{\mathbf{T}}_l = \begin{pmatrix} 1 & 2 & \infty \\ 1 & 1 & 1 \\ 1 & 1 & \infty \end{pmatrix} \tag{7}$$

and vectors $\tilde{\mathbf{a}} = (0, \infty, \infty)$, $\tilde{\mathbf{b}} = (\infty, 3, 1)$. The weight function is chosen such that one element per equivalence class receives weight 0 and the remaining receive weight ∞. For different transition sequences we obtain the following automaton behavior.

seq	l^0	l^1	l^2	l^3	l^4
$\tilde{\mathbf{d}}_{seq}$	$\begin{pmatrix} 0 \\ \infty \\ \infty \end{pmatrix}$	$\begin{pmatrix} 1 \\ 2 \\ \infty \end{pmatrix}$	$\begin{pmatrix} 2 \\ 3 \\ 3 \end{pmatrix}$	$\begin{pmatrix} 3 \\ 4 \\ 4 \end{pmatrix}$	$\begin{pmatrix} 4 \\ 5 \\ 5 \end{pmatrix}$.
$\tilde{c}(seq)$	∞	5	4	5	6

Like bisimulation for labeled automata, bisimulation for weighted automata can be characterized from below by building unions of bisimulations and also from above by a partition refinement algorithm. The latter only works for finite branching as it is the case for the automata model proposed here.

Theorem 3.5. *Let $\mathbf{V}_1$ and $\mathbf{V}_2$ be two bisimulation relations for automaton $\mathcal{A}$. Let $\mathcal{E}^1$ and $\mathcal{E}^2$ be the corresponding equivalence relations. Then also $\mathcal{E}^0 = \mathcal{E}^1 \cup^* \mathcal{E}^2$ defines a bisimulation where $\mathcal{E}^1 \cup^* \mathcal{E}^2$ is the least equivalence relation containing $\mathcal{E}^1 \cup \mathcal{E}^2$.*

Proof. We have to show that the quotient of $\mathcal{E}^0$ is a surjective functional simulation. Since $\mathcal{E}^0$ is an equivalence relation, the quotient is a surjective function with relational matrix $\mathbf{V}_0$. Let $\mathcal{A} \longrightarrow \widetilde{\mathcal{A}}$ according to relational matrix $\mathbf{V}_0$. It remains to show that it is a functional simulation. We have to show $\mathbf{b} = \mathbf{V}_0\tilde{\mathbf{b}}$ and $\mathbf{T}_l\mathbf{V}_0 = \mathbf{V}_0\tilde{\mathbf{T}}_l$.

Since $\mathbf{b}(x) = \mathbf{b}(y)$ for $x, y \in \mathcal{E}^i_z$ for $i \in \{1, 2\}$ and some equivalence class z, the condition for vector $\mathbf{b}$ holds.

Each equivalence class of $\mathcal{E}^0_z$ can be expressed as

$$\mathcal{E}^0_z = \mathcal{E}^1_{x_1} \cup \ldots \cup \mathcal{E}^1_{x_R} = \mathcal{E}^2_{y_1} \cup \ldots \cup \mathcal{E}^2_{y_S}$$

Furthermore, for $x, y \in \mathcal{E}^0_z$ either $x, y \in \mathcal{E}^1_{z_1}$ or $x, y \in \mathcal{E}^2_{z_2}$ holds. Without loss of generality we assume that $x, y \in \mathcal{E}^1_{z_1}$ such that for all $\mathcal{E}^0_z$ the following identity holds.

$$\begin{aligned}\widehat{\sum}_{v\in\mathcal{E}^0_z}\mathbf{T}_l(x, v) &= \widehat{\sum}_{r=1}^{R}\widehat{\sum}_{v\in\mathcal{E}^1_{x_r}}\mathbf{T}_l(x, v)\\ &= \widehat{\sum}_{r=1}^{R}\widehat{\sum}_{v\in\mathcal{E}^1_{x_r}}\mathbf{T}_l(y, v) = \widehat{\sum}_{v\in\mathcal{E}^0_z}\mathbf{T}_l(y, v).\end{aligned}$$

By setting $\tilde{\mathbf{T}}_l(v, w) = \widehat{\sum}_{y\in\mathcal{E}^0_w}\mathbf{T}_l(x, y)$ the required relation follows. □

The theorem implies that a largest bisimulation exists and is given as the union of all bisimulations for automaton $\mathcal{A}$. The largest bisimulation is unique up to the numbering of equivalence classes and yields the smallest aggregated automaton that is bisimilar to automaton $\mathcal{A}$.

Corollary 3.6. *Two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are bisimulation equivalent, iff an automaton $\mathcal{A}$ and surjective functional simulations $\mathcal{A}_1 \longrightarrow \mathcal{A}$ and $\mathcal{A}_2 \longrightarrow \mathcal{A}$ exist.*

We denote bisimulation equivalence of automata $\mathcal{A}_1$ and $\mathcal{A}_2$ as $\mathcal{A}_1 \approx \mathcal{A}_2$. For an effective computation of the largest bisimulation, the following characterization of bisimulations from above is more helpful.

Theorem 3.7. *For an automaton $\mathcal{A}$ let $\mathcal{E}^0, \mathcal{E}^1, \ldots$ be a family of equivalence relations and $\mathbf{V}^1$, $\mathbf{V}^2$, $\ldots$ be the corresponding quotient maps that are defined as follows:*

(1) $\mathcal{E}^0 = \{(x, y)|\mathbf{b}(x) = \mathbf{b}(y)\}$,
(2) $\mathcal{E}^{r+1} = \{(x, y)|(x, y) \in \mathcal{E}^r \wedge \forall l \in Act : \mathbf{e}_x\mathbf{T}_l\mathbf{V}^r = \mathbf{e}_y\mathbf{T}_l\mathbf{V}^r\}$.

If $\mathcal{E}^r = \mathcal{E}^{r+1}$, then $\mathcal{E}^r = \mathcal{E}^{r+s} = \mathcal{E}^*$ for all $s > 0$. $\mathcal{E}^*$ is the largest bisimulation.

Proof. It is easy to show that $\mathcal{E}^r = \mathcal{E}^{r+1}$ implies that the fixed point of the refinement has been reached and that $\mathcal{E}^*$ is a bisimulation. To prove that $\mathcal{E}^*$ is the largest bisimulation, we consider two states x and y that are bisimilar according to some bisimulation $\mathcal{E}$. We prove that x and y will never be put into different equivalence classes. Obviously all states that are bisimilar according to an arbitrary bisimulation relation have to be in the same equivalence class of $\mathcal{E}^0$ because $\mathbf{b}(x) = \mathbf{b}(y)$ is a necessary condition for bisimilar states. Thus, each equivalence class of $\mathcal{E}^0$ is built from the union of equivalence classes of an arbitrary bisimulation relation $\mathcal{E}$. Using the relation applied in the proof of Theorem 3.5, an equivalence class of $\mathcal{E}^{r+1}$ is built from the union of equivalence classes of $\mathcal{E}$, if this is true for the equivalence classes of $\mathcal{E}^r$. By induction over r it is easy to show that each equivalence class of $\mathcal{E}^*$ is built from the union of equivalence classes of $\mathcal{E}$ such that x and y have to be in the same equivalence class of each $\mathcal{E}^r$ and of $\mathcal{E}^*$. □

The partition refinement approach defined in the theorem can be used for the formulation of an algorithm to compute bisimulation relations for finite state systems as done for untimed automata [19] for a long time and more recently also for probabilistic or stochastic systems [4,13]. The mentioned algorithms have a complexity of $O(nt)$ to compute the largest bisimulation for an automaton with n states and t transitions and can be easily extended to handle transition weights from an arbitrary semiring instead of rates or probabilities. Thus, for an automaton, the largest bisimulation can be computed and the corresponding aggregated automaton can be generated. For untimed automata [38] and more recently also for stochastic models [21] more efficient algorithms with complexity $O(n \log t)$ have been proposed. These algorithms are based on some preordering of weights which implies that the used semiring has to observe specific conditions like a complete ordering.

In the sequel we denote the aggregated automaton that has been generated for automata $\mathcal{A}$ according to the largest bisimulation as $\widetilde{\mathcal{A}}^*$.

Theorem 3.8. *The relation $\mathcal{A}_1 + \mathcal{A}_2 \approx \mathcal{A}_2 + \mathcal{A}_1$ holds for arbitrary semirings. If the multiplication of the semiring is commutative, then $\mathcal{A}_1 \|_{Act_c} \mathcal{A}_2 \approx \mathcal{A}_2 \|_{Act_c} \mathcal{A}_1$ for all $Act_c \subseteq Act_1 \cup Act_2$.*

Proof. The proof follows from a simple permutation of the states. □

Bisimulation allows one to reduce the size of automata by representing classes of equivalent states by a single state while the behavior of the complete automaton is preserved. However, to be really useful, the equivalence property has to be preserved by composition operations such that the reduction approach can be applied compositionally. The following theorem shows that bisimulation is indeed preserved by the composition operations defined above, i.e., it is a congruence according to these operations.

Theorem 3.9. *If $\mathcal{A}_1 \approx \mathcal{A}_2$, then*

(1) $\mathcal{A}_1 + \mathcal{A}_3 \approx \mathcal{A}_2 + \mathcal{A}_3$,
(2) $\mathcal{A}_1 \cdot \mathcal{A}_3 \approx \mathcal{A}_2 \cdot \mathcal{A}_3$,
(3) $\mathcal{A}_3 \cdot \mathcal{A}_1 \approx \mathcal{A}_3 \cdot \mathcal{A}_2$, *and*
(4) *for all $Act_c \subseteq Act_1 \cap Act_3$: $\mathcal{A}_1 \|_{Act_c} \mathcal{A}_3 \approx \mathcal{A}_2 \|_{Act_c} \mathcal{A}_3$*

for some automaton $\mathcal{A}_3$.

Proof. By Corollary 3.6 it is sufficient to show that a surjective functional simulation $\mathbf{V}^{(i)}$, $\mathcal{A}_i \longrightarrow \mathcal{A}_*$ $(i = 1, 2)$ implies the existence of surjective functional simulations

(1) $\mathcal{A}_i + \mathcal{A}_3 \longrightarrow \mathcal{A}_* + \mathcal{A}_3$,
(2) $\mathcal{A}_i \cdot \mathcal{A}_3 \longrightarrow \mathcal{A}_* \cdot \mathcal{A}_3$,
(3) $\mathcal{A}_3 \cdot \mathcal{A}_i \longrightarrow \mathcal{A}_3 \cdot \mathcal{A}_*$, and
(4) $\mathcal{A}_i \|_{Act_c} \mathcal{A}_3 \longrightarrow \mathcal{A}_* \|_{Act_c} \mathcal{A}_3$.

We denote the automata resulting from the composition by $\mathcal{A}_{13}$ and $\mathcal{A}_{23}$, respectively. The automaton on the right side is denoted as $\mathcal{A}_{*3}$.

In the first three cases the relation $\mathbf{V}^{(i3)}$ $\mathcal{A}_{i3} \longrightarrow \mathcal{A}_{*3}$ is defined as

$$\mathbf{V}^{(i3)} = \begin{pmatrix} \mathbf{V}^{(i)} & \mathbb{O} \\ \mathbb{O} & \mathbf{I} \end{pmatrix}$$

where $\mathbf{I}$ is an identity matrix of a dimension that equals the number of states of $\mathcal{A}_3$. The relation is a surjective function since $\mathbf{V}^{(i)}$ and $\mathbf{I}$ are surjective functions. The conditions $\mathbf{b}^{(i3)} = \mathbf{V}^{(i3)}\mathbf{b}^{(*3)}$ and $\mathbf{T}_l^{(i3)}\mathbf{V}^{(i3)} = \mathbf{V}^{(i3)}\mathbf{T}_l^{(*3)}$ follow by simple matrix calculations (see also [6, Chap 9]).

For the synchronized product $\mathbf{V}^{(i3)} = \mathbf{V}^{(i)} \widehat{\otimes} \mathbf{I}$. $\mathbf{V}^{(i3)}$ is a surjective function if $\mathbf{V}^{(i)}$ is a surjective function. Furthermore we have

$$\begin{aligned}
&\left(\mathbf{a}^{(i)} \widehat{\otimes} \mathbf{a}^{(3)}\right)\left(\mathbf{V}^{(i)} \widehat{\otimes} \mathbf{I}\right) = \mathbf{a}^{(i)}\mathbf{V}^{(i)} \widehat{\otimes} \mathbf{a}^{(3)} = \mathbf{a}^{(*)} \widehat{\otimes} \mathbf{a}^{(3)},\\
&\left(\mathbf{V}^{(i)} \widehat{\otimes} \mathbf{I}\right)\left(\mathbf{b}^{(i)} \widehat{\otimes} \mathbf{b}^{(3)}\right) = \mathbf{V}^{(i)}\mathbf{b}^{(i)} \widehat{\oplus} \mathbf{b}^{(3)} = \mathbf{b}^{(*)} \widehat{\oplus} \mathbf{b}^{(3)},\\
&\left(\mathbf{T}_l^{(i)} \widehat{\otimes} \mathbf{T}_l^{(3)}\right)\left(\mathbf{V}^{(i)} \widehat{\otimes} \mathbf{I}\right) = \mathbf{T}_l^{(i)}\mathbf{V}^{(i)} \widehat{\otimes} \mathbf{T}_l^{(3)} = \left(\mathbf{V}^{(i)} \widehat{\otimes} \mathbf{I}\right)\left(\mathbf{T}_l^{(*)} \widehat{\otimes} \mathbf{T}_l^{(3)}\right)
\end{aligned}$$

and

$$\begin{aligned}
&\left(\mathbf{T}_l^{(i)} \widehat{\oplus} \mathbf{T}_l^{(3)}\right)\left(\mathbf{V}^{(i)} \widehat{\otimes} \mathbf{I}\right) = \left(\mathbf{T}_l^{(i)} \widehat{\otimes} \mathbf{I} \widehat{+} \mathbf{I} \widehat{\otimes} \mathbf{T}_l^{(3)}\right)\left(\mathbf{V}^{(i)} \widehat{\otimes} \mathbf{I}\right) =\\
&\mathbf{T}_l^{(i)}\mathbf{V}^{(i)} \widehat{\otimes} \mathbf{I} \widehat{+} \mathbf{V}^{(i)} \widehat{\otimes} \mathbf{T}_l^{(3)} = \left(\mathbf{V}^{(i)} \widehat{\otimes} \mathbf{I}\right)\left(\mathbf{T}_l^{(*)} \widehat{\otimes} \mathbf{I} \widehat{+} \mathbf{I} \widehat{\otimes} \mathbf{T}_l^{(3)}\right) =\\
&\left(\mathbf{V}^{(i)} \widehat{\otimes} \mathbf{I}\right)\left(\mathbf{T}_l^{(*)} \widehat{\oplus} \mathbf{T}_l^{(3)}\right)
\end{aligned}$$

such that $\mathbf{V}^{(i3)}$ is a functional simulation $\mathcal{A}_{i3} \longrightarrow \mathcal{A}_{*3}$. □

4. Backward bisimulation for weighted automata

Bisimulation considers the future behavior of an automaton by defining states as equivalent if they yield equivalent successors. As shown in [11,12] one can also define a bisimulation relation for stochastic and untimed automata that considers the past behavior by defining states as equivalent, if they can be reached from equivalent origins by equivalent rates. The resulting equivalence relation has been denoted as exact performance equivalence. It has been shown in the mentioned papers that the resulting equivalence preserves several interesting aspects of the behavior of stochastic systems. Furthermore, forward and backward bisimulation yield different equivalence relations. With respect to state space reduction by aggregating of equivalent states, sometimes forward and sometimes backward bisimulation might be advantageous. For stochastic systems the relation between backward bisimulation and the stochastic logic CSL has been recently investigated [41]. It is known that forward bisimilar states cannot be distinguished by CSL formulas [5], whereas backward bisimilar states can be distinguished under some CSL formulas. However, a complete logical characterization of backward bisimulation in the stochastic setting is still missing. In the sequel of this section we introduce two different forms of backward bisimulation for weighted automata which extend the relation for untimed and stochastic automata and show that these relations preserve several results and are congruence relations according to the composition operations. Since backward bisimulations consider the past behavior we define first the transposed automaton where the direction of transitions is reversed.

Definition 4.1. For a weighted automaton $\mathcal{A} = (\mathbf{a}, (\mathbf{T}_l)_{l \in Act}, \mathbf{b})$, the transposed automaton is defined as $\mathcal{A}^T = (\mathbf{b}^T, ((\mathbf{T}_l)^T)_{l \in Act}, \mathbf{a}^T)$.

Like bisimulation, backward bisimulation is characterized using the matrices and vectors of the automaton description. We first introduce a corollary which follows by simple matrix calculations from our results on bisimulation.

Corollary 4.1. *If $\mathbf{V}$ is a relational simulation $\mathcal{A}_1 \longrightarrow \mathcal{A}_2$, then $\mathbf{V}^T$ is a relational simulation $\mathcal{A}_2^T \longrightarrow \mathcal{A}_1^T$.*

Definition 4.2. Let $\mathcal{A}$ be an automaton with n states and let $\mathbf{V}$ be a surjective function between $[n]$ and $[m]$, then $\mathbf{V}$ is a backward bisimulation of type 1 if it is a relational simulation $\mathcal{A}^T \longrightarrow \widetilde{\mathcal{A}}^T$ where $\widetilde{\mathcal{A}}$ is the aggregated automaton generated as in Definition 3.3 for some appropriate weight vector from automaton $\mathcal{A}^T$.

Since the aggregated automaton according to a backward bisimulation of type 1 is computed on the transposed matrices, it is invariant under different weight vectors. In contrast to the conditions for bisimulation which are based on outgoing transitions and state valuation, we now have conditions on incoming transitions and the initialization function. As before the quotient map of a backward bisimulation of type 1 is an equivalence relation on $[n]$.

Theorem 4.2. *Let $\widetilde{\mathcal{A}}$ be an aggregated automaton resulting from automaton $\mathcal{A}$ according to some backward bisimulation $\mathbf{V}$ of type 1, then*
(1) *$\mathcal{A}$ and $\widetilde{\mathcal{A}}$ are behaviorally equivalent,*
(2) *$\mathbf{d}_{seq}(x) = \tilde{\mathbf{d}}_{seq}(z)$ for all $z \in [m]$ and $x \in \mathcal{E}_z$ where $\mathcal{E}$ is the equivalence relation belonging to $\mathbf{V}$.*

Proof. Since $\mathbf{V}$ is a bisimulation $\mathcal{A}^T \rightarrow \widetilde{\mathcal{A}}^T$, we have

$$\begin{array}{lll} (\mathbf{a})^T = \mathbf{V}(\tilde{\mathbf{a}})^T & \Leftrightarrow ((\mathbf{a})^T)^T = (\mathbf{V}(\tilde{\mathbf{a}})^T)^T & \Leftrightarrow \mathbf{a} = \tilde{\mathbf{a}}\mathbf{V}^T \\ (\mathbf{T}_l)^T\mathbf{V} = \mathbf{V}(\tilde{\mathbf{T}}_l)^T & \Leftrightarrow ((\mathbf{T}_l)^T\mathbf{V})^T = (\mathbf{V}(\tilde{\mathbf{T}}_l)^T)^T & \Leftrightarrow \mathbf{V}^T\mathbf{T}_l = \tilde{\mathbf{T}}_l\mathbf{V}^T \\ (\mathbf{b})^T\mathbf{V} = \tilde{\mathbf{b}} & \Leftrightarrow ((\mathbf{b})^T\mathbf{V})^T & \Leftrightarrow \tilde{\mathbf{b}} = \mathbf{V}^T\mathbf{b}. \end{array}$$

Observe that the above equalities also hold for non-commutative semirings since matrix $\mathbf{V}$ contains only $\mathbb{O}$ and $\mathbb{1}$ as elements.

We first prove 2.

$$\begin{array}{llll} \mathbf{d}_{seq} & = \mathbf{a} \,\widehat{\cdot}\, \widehat{\prod}_{i=1}^{k} \mathbf{T}_{l_i} & = \tilde{\mathbf{a}} \,\widehat{\cdot}\, \mathbf{V}^T \,\widehat{\cdot}\, \widehat{\prod}_{i=1}^{k} \mathbf{T}_{l_i} & \\ & = \tilde{\mathbf{a}} \,\widehat{\cdot}\, \tilde{\mathbf{T}}_{l_1} \,\widehat{\cdot}\, \mathbf{V}^T \,\widehat{\cdot}\, \widehat{\prod}_{i=2}^{k} \mathbf{T}_{l_i} & = \ldots & = \tilde{\mathbf{d}}_{seq} \,\widehat{\cdot}\, \mathbf{V}^T. \end{array}$$

Since $\mathbf{V}(x, z) = \mathbb{1}$ for $x \in \mathcal{E}_z$, $\mathbf{d}_{seq}(x) = \tilde{\mathbf{d}}_{seq}(z)$. For 1. we have

$$c(seq) = \mathbf{d}_{seq} \,\widehat{\cdot}\, \mathbf{b} = \tilde{\mathbf{d}}_{seq} \,\widehat{\cdot}\, \mathbf{V}^T \,\widehat{\cdot}\, \mathbf{b} = \tilde{\mathbf{d}}_{seq} \,\widehat{\cdot}\, \tilde{\mathbf{b}} = \tilde{c}(seq). \quad \square$$

A second variant of backward bisimulation can be defined by generalizing the concept of weak lumpability for Markov chains [33,36]. The resulting bisimulation is based on matrix $\mathbf{W}$ rather than $\mathbf{V}$.

Definition 4.3. Let $\mathcal{A}$ be an automaton with n states and $\widetilde{\mathcal{A}}$ the aggregated automaton with m states according to a pair of matrices $\mathbf{V}$ and $\mathbf{W}$ (cf. Definition 3.3), then $\mathbf{V}, \mathbf{W}$ is a backward bisimulation of type 2 iff

$$\mathbf{V} \text{ is a surjective function from } [n] \text{ to } [m] \text{ and } \mathbf{a} = \tilde{\mathbf{a}}\mathbf{W},\ \ \mathbf{W}\mathbf{T}_l = \tilde{\mathbf{T}}_l\mathbf{W},\ \ \mathbf{W}\mathbf{b} = \tilde{\mathbf{b}}.$$

Like backward bisimulations of type 1, also backward bisimulations of type 2 put some restrictions on the matrices $\mathbf{T}_l$ and vector $\mathbf{a}$ but not on $\mathbf{b}$. However, in contrast to forward bisimulation and backward bisimulation of type 1, backward bisimulation of type 2 cannot be constructed using a partition refinement approach because it depends on the weight vector.

Theorem 4.3. *Let $\widetilde{\mathcal{A}}$ be an aggregated automaton resulting from automaton $\mathcal{A}$ according to some backward bisimulation* $\mathbf{V}, \mathbf{W}$ *of type* 2*, then*

(1) *$\mathcal{A}$ and $\widetilde{\mathcal{A}}$ are behaviorally equivalent,*
(2) *$\mathbf{d}_{seq}(x) = \tilde{\mathbf{d}}_{seq}(z) \widehat{\cdot} \mathbf{W}(z, x)$ for all $z \in [m]$ and $x \in \mathcal{E}_z$ where $\mathcal{E}$ is the equivalence relation belonging to* $\mathbf{V}$.

Proof. The proof is similar to the proof of Theorem 4.2. We first prove 2.

$$\begin{aligned} \mathbf{d}_{seq} &= \mathbf{a} \widehat{\cdot} \widehat{\prod}_{i=1}^{k} \mathbf{T}_{l_i} && = \tilde{\mathbf{a}} \widehat{\cdot} \mathbf{W} \widehat{\cdot} \widehat{\prod}_{i=1}^{k} \mathbf{T}_{l_i} \\ &= \tilde{\mathbf{a}} \widehat{\cdot} \tilde{\mathbf{T}}_{l_1} \widehat{\cdot} \mathbf{W} \widehat{\cdot} \widehat{\prod}_{i=2}^{k} \mathbf{T}_{l_i} && = \ldots \qquad = \tilde{\mathbf{d}}_{seq} \widehat{\cdot} \mathbf{W}. \end{aligned}$$

For 1. we have

$$c(seq) = \mathbf{d}_{seq} \widehat{\cdot} \mathbf{b} = \tilde{\mathbf{d}}_{seq} \widehat{\cdot} \mathbf{W} \widehat{\cdot} \mathbf{b} = \tilde{\mathbf{d}}_{seq} \widehat{\cdot} \tilde{\mathbf{b}} = \tilde{c}(seq). \quad \square$$

Observe that both variants of backward bisimulation allow the recreation of $\mathbf{d}_{seq}$ from $\tilde{\mathbf{d}}_{seq}$. This is not possible for forward bisimulation which shows that the backward versions are stronger with respect to result preservation. However, aggregation according to backward bisimulation is not transition value preserving.

Corollary 4.4. *Two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are backward bisimulation equivalent of type i, iff an automaton $\mathcal{A}$ and backward bisimulations of type i with $\mathcal{A}_1 \longrightarrow \mathcal{A}$ and $\mathcal{A}_2 \longrightarrow \mathcal{A}$ exist.*

For backward bisimulation equivalent automata $\mathcal{A}_1$ and $\mathcal{A}_2$ according to some backward bisimulation of type i, we use the notation $\mathcal{A}_1 \simeq_i \mathcal{A}_2$.

Theorem 4.5. *If $\mathcal{A}_1 \simeq_i \mathcal{A}_2$ ($i = 1, 2$), then*

(1) *$\mathcal{A}_1 + \mathcal{A}_3 \simeq_i \mathcal{A}_2 + \mathcal{A}_3$,*
(2) *$\mathcal{A}_1 \cdot \mathcal{A}_3 \simeq_i \mathcal{A}_2 \cdot \mathcal{A}_3$,*
(3) *$\mathcal{A}_3 \cdot \mathcal{A}_1 \simeq_i \mathcal{A}_3 \cdot \mathcal{A}_2$, and*
(4) *for all $Act_c \subseteq Act_1 \cap Act_3$: $\mathcal{A}_1 \|_{Act_c} \mathcal{A}_3 \simeq_i \mathcal{A}_2 \|_{Act_c} \mathcal{A}_3$*

for some automaton $\mathcal{A}_3$.

Proof. The proof is similar to the proof for Theorem 3.9. Properties have to be proved using the matrices $\mathbf{V}^{(i3)}$ and $\mathbf{W}^{(i3)}$ ($i = 1, 2$) for the composed automaton. For the direct sum and product we define

$$\mathbf{V}^{(i3)} = \begin{pmatrix} \mathbf{V}^{(i)} & \mathbb{O} \\ \mathbb{O} & \mathbf{I} \end{pmatrix}, \qquad \mathbf{W}^{(i3)} = \begin{pmatrix} \mathbf{W}^{(i)} & \mathbb{O} \\ \mathbb{O} & \mathbf{I} \end{pmatrix}$$

and for the synchronized product $\mathbf{V}^{(i3)} = \mathbf{V}^{(i)} \widehat{\otimes} \mathbf{I}$, $\mathbf{W}^{(i3)} = \mathbf{W}^{(i)} \widehat{\otimes} \mathbf{I}$. The proof of the properties required for backward bisimulation follows by simple matrix calculations. $\square$

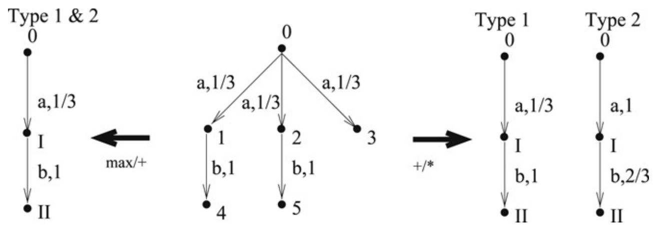

Fig. 2. A probabilistic automaton $\mathcal{A}$ and the aggregated automata $\widetilde{\mathcal{A}}$.

Examples: We present some examples to show the differences between the variants of bisimulation.

Fig. 2 shows the probabilistic process which has already been presented in Fig. 1. The largest backward bisimulation of type 1 and 2 is defined with the relation

$$\mathbf{V} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}^T .$$

For type 1 we obtain the first aggregated process on the right side. The corresponding weight vector is $\tilde{\mathbf{b}} = (0, 1, 2)^T$. The second process on the right side is the aggregated process for type 2 built with weight vector $\mathbf{w} = (1, 1/3, 1/3, 1/3, 1/2, 1/2)$. The corresponding vector $\tilde{\mathbf{b}}$ equals $(0, 1/3, 1)$. The resulting aggregations are not transition value preserving since the probabilities of outgoing transitions sum up to values between 0 and 1. Thus, the aggregated automata do not belong to the class of probabilistic automata where outgoing transition probabilities sum up to 0 or 1. Nevertheless, the behavior of both automata, entire and aggregated one, is the same. The difference is that another interpretation of automata behavior is required. In the entire model an automaton can perform a transition of a specific type or it cannot perform it. In the aggregated model an automaton can perform a transition with some probability p and with probability $1 - p$ it cannot perform the transition.

If we interpret the automaton over semiring $([0, 1], \max, \cdot, 0, 1)$, the largest backward bisimulation of both types can be built with the same relation. For type 2, the weight vector has to include at least one element equal to 1 for a state of each equivalence class such that corresponding aggregated automata for backward bisimulation of type 1 and 2 have to be identical and are shown on the left side of Fig. 2. For this semiring $\tilde{\mathbf{b}} = (0, 1, 1)^T$.

As a last example we consider the automaton over semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ shown in (6). The largest backward bisimulation of type 1 and 2 for this automaton and the corresponding aggregated matrix equal

$$\mathbf{V} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}^T \quad \text{and} \quad \tilde{\mathbf{T}}_l = \begin{pmatrix} 1 & 2 & \infty \\ \infty & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \tag{8}$$

The vectors are $\tilde{\mathbf{a}} = (0, \infty, \infty)$ and $\tilde{\mathbf{b}} = (\infty, 3, 1)$. The weight vector has to be chosen such that all elements are equal to 0 to assure that $\min_{x \in \mathcal{E}_z} \mathbf{w}(x) = 0 = \mathbb{1}$. We obtain the following results for the automaton.

seq	l^0	l^1	l^2	l^3	l^4
$\tilde{\mathbf{d}}_{seq}$	$\begin{pmatrix} 0 \\ \infty \\ \infty \end{pmatrix}$	$\begin{pmatrix} 1 \\ 2 \\ \infty \end{pmatrix}$	$\begin{pmatrix} 2 \\ 3 \\ 3 \end{pmatrix}$	$\begin{pmatrix} 3 \\ 4 \\ 4 \end{pmatrix}$	$\begin{pmatrix} 4 \\ 5 \\ 5 \end{pmatrix}$.
$\tilde{c}(seq)$	∞	5	4	5	6

The behavior is identical to the behavior of the aggregated automaton according to bisimulation as defined in (7). However, now we have the additional result $\mathbf{d}_{seq}(x) = \min_{y \in \mathcal{E}_z} \mathbf{d}_{seq}(y) = \tilde{\mathbf{d}}_{seq}(z)$ which implies $\mathbf{d}_{seq}(x) = \mathbf{d}_{seq}(y)$ for $x, y \in \mathcal{E}_z$.

5. Concluding remarks and future work

In this paper we present a general automata model which covers different well known types of automata with transition labeling and valuation. For this automata model, which is a slight extension of weighted automata [34], we define forward and backward bisimulation and show that these relations preserve main characteristics of the behavior. The proposed bisimulation relations cover known bisimulation relations for different types of automata like probabilistic or stochastic automata and they naturally define bisimulation for new classes of models like for max/plus systems. Furthermore, it is shown that the bisimulation relations are congruence relations according to several natural composition operations for weighted automata. Bisimulations can be used to compute for a given automaton an aggregated automaton with an equivalent behavior and they can as well be used to decide whether two automata are equivalent.

Several aspects will be considered in future work. First, other equivalence or preorder relations can be defined for this general class of automata and may give a general framework for defining equivalence of discrete systems. Another aspect which should be considered in the future is the relation between bisimulation and logics. It is well known that bisimilar untimed automata are indistinguishable under formulas of Hennessy, Milner logic [26]. Other logics have been defined for probabilistic or stochastic systems and have been used for model checking of those models [2,5,35]. It should be possible to define a general logic for the automata model presented in this paper which covers known approaches and allows us to extend the idea of model checking to a much more general framework. A first attempt in this direction has been made in [17].

Acknowledgments

I thank the anonymous reviewers and the editor for their very detailed reviews and hints that helped to improve the presentation significantly and also resulted in a much more general approach to defining and presenting backward bisimulation.

References

[1] M. Ajmone-Marsan, G. Balbo, G. Conte, A class of generalized stochastic Petri nets for performance evaluation of multiprocessor systems, ACM Transactions on Computer Systems 2 (2) (1984) 93–122.
[2] A. Aziz, K. Sanwal, V. Singhai, R. Brayton, Verifying continuous time Markov chains, in: Computer Aided Verification'97, in: LNCS, vol. 1102, Springer, 1997, pp. 269–278.
[3] F. Baccelli, G. Cohen, G. Olsder, J. Quadrat, Synchronization and Linearity, John Wiley and Sons, 1992.
[4] C. Baier, B. Engelen, M. Majster-Cederbaum, Deciding bisimilarity and similarity for probabilistic processes, Journal of Computer and System Sciences 60 (1) (2000) 187–231.
[5] C. Baier, B. Haverkort, H. Hermanns, J.P. Katoen, Model checking algorithms for continuous time Markov chains, IEEE Transactions on Software Engineering 29 (7) (2003) 524–541.
[6] S.L. Bloom, Z. Esik, Iteration theories, in: EATCS Monographs on Theoretical Computer Science, Springer, 1993.
[7] R. Blute, J. Deshrnais, A. Edalat, R. Panangaden, Bisimulation for labelled Markov processes, in: 12th Annual IEEE Symposium on Logic in Computer Science, IEEE CS Press, 1997, pp. 149–158.
[8] P. Buchholz, Markovian process algebra: composition and equivalence, in: U. Herzog, M. Rettelbach (Eds.), Proc. of the 2nd Work. on Process Algebras and Performance Modelling, in: Arbeitsberichte des IMMD, vol. 27, University of Erlangen, 1994, pp. 11–30.
[9] P. Buchholz, Equivalence relations for stochastic automata networks, in: W.J. Stewart (Ed.), Computations with Markov Chains, Kluwer Academic Publishers, 1995, pp. 197–216.
[10] P. Buchholz, A notion of equivalence for stochastic Petri nets, in: G. De Michelis, M. Diaz (Eds.), Application and Theory of Petri Nets 1995, in: LNCS, vol. 935, Springer, 1995, pp. 161–180.
[11] P. Buchholz, Exact performance equivalence — an equivalence relation for stochastic automata, Theoretical Computer Science 215 (1–2) (1999) 263–287.
[12] P. Buchholz, Hierarchical structuring of superposed GSPNs, IEEE Transactions on Software Engineering 25 (2) (1999) 166–181.
[13] P. Buchholz, Efficient computation of equivalent and reduced representations for stochastic automata, Int. Journ. Computer Systems Science and Engineering 15 (2) (2000) 93–103.
[14] P. Buchholz, P. Kemper, Quantifying the dynamic behavior of process algebras, in: L. de Alfaro, S. Gilmore (Eds.), Process Algebras and Probabilistic Methods, in: LNCS, vol. 2165, Springer, 2001, pp. 184–199.
[15] P. Buchholz, P. Kemper, Efficient computation and representation of large reachability sets for composed automata, Discrete Event Dynamic Systems Theory and Applications 12 (3) (2002) 265–286.
[16] P. Buchholz, P. Kemper, Weak bisimulation for (max/+) automata and related models, Journal of Automata, Languages and Combinatorics 8 (2) (2003) 187–218.
[17] P. Buchholz, P. Kemper, Model checking for automata with transition costs, (2006) (submitted for publication).
[18] G. Ciardo, A.S. Miner, S. Donatelli, Using the exact state space of a model to compute approximate stationary measures, in: J. Kurose, P. Nain (Eds.), Proc. ACM Sigmetrics, ACM Press, 2000, pp. 207–216.
[19] R. Cleaveland, J. Parrow, B. Steffen, The concurrency workbench: A semantics based tool for the verification of concurrent systems, ACM Transactions on Programming Languages and Systems 15 (1) (1993) 36–72.

[20] E.P. de Vink, J.J.M.M. Rutten, Bisimulation for probabilistic transition systems: A coalgebraic approach, in: P. Degano, R. Gorrieri, A. Marchetti-Spaccamela (Eds.), Proc. ICALP'97, in: LNCS, vol. 1256, Springer, 1997, pp. 460–470.
[21] S. Derisavi, H. Hermanns, W.H. Sanders, Optimal state space lumping in Markov chains, Information Processing Letters 54 (2) (2003) 309–315.
[22] J. Desharnais, A. Edalat, R. Panangaden, Bisimulation for labelled Markov processes, Information and Computation 179 (2) (2002) 163–193.
[23] S. Eilenberg, Automata, Languages and Machines Part A, in: Pure and Applied Mathematics: A Series of Monographs and Textbook, vol. 58, Academic Press, 1974.
[24] S. Gaubert, Performance evaluation of (max/+) automata, IEEE Transactions on Automatic Control 40 (12) (1995) 2014–2025.
[25] S. Graf, B. Steffen, G. Lüttgen, Compositional minimization of finite state systems, Formal Aspects of Computing 8 (5) (1996) 607–616.
[26] M.C. Hennessy, R. Milner, Algebraic laws for non-determinism and concurrency, Journal of the ACM 32 (1985) 137–161.
[27] H. Hermanns, U. Herzog, V. Mertsiotakis, Stochastic process algebras — between LOTOS and Markov chains, Computer Networks and ISDN Systems 30 (9–10) (1998) 901–924.
[28] H. Hermanns, M. Rettelbach, Syntax, semantics, equivalences, and axioms for MTIPP, in: U. Herzog, M. Rettelbach (Eds.), Proc. of the 2nd Work. on Process Algebras and Performance Modelling, in: Arbeitsberichte des IMMD, vol. 27, University of Erlangen, 1994.
[29] J. Hillston, A compositional approach for performance modelling. Ph.D. Thesis, University of Edinburgh, Dep. of Comp. Sc., 1994.
[30] J. Hillston, The nature of synchronization, in: U. Herzog, M. Rettelbach (Eds.), Proc. of the 2nd Workshop on Process Algebra and Performance Modeling, in: Arbeitsberichte des IMMD, vol. 27, University of Erlangen, 1994, pp. 51–70.
[31] J. Hillston, Compositional Markovian modelling using a process algebra, in: W.J. Stewart (Ed.), Computations with Markov Chains, Kluwer Academic Publishers, 1995, pp. 177–196.
[32] C.C. Jou, S. Smolka, Equivalences, congruences and complete axiomatizations for probabilistic processes, in: J.C.M. Baeten, J.W. Klop (Eds.), Proc. CONCUR'90, in: LNCS, vol. 458, Springer, 1990, pp. 367–383.
[33] J.G. Kemeny, J.L. Snell, Finite Markov Chains, Springer, 1976.
[34] W. Kuich, A. Salomaa, Semirings, Automata, Languages, in: ETACS Monographs on Theoretical Computer Science, Springer, 1986.
[35] K. Larsen, A. Skou, Bisimulation through probabilistic testing, Information and Computation 94 (1991) 1–28.
[36] J. Ledoux, Weak lumpability of finite Markov chains and positive invariance of cones, Technical Report RR-2801, INRIA, 1996.
[37] R. Milner, Communication and Concurrency, Prentice Hall, 1989.
[38] R. Paige, R.E. Tarjan, Three partition refinement algorithms, SIAM Journal on Computing 16 (6) (1987) 973–989.
[39] D. Park, Concurrency and automata on infinite sequences, in: Proc. 5th GI Conference on Theoretical Computer Science, in: LNCS, vol. 104, Springer, 1981, pp. 167–183.
[40] B. Plateau, On the stochastic structure of parallelism and synchronisation models for distributed algorithms, ACM Performance Evaluation Review 13 (1985) 142–154.
[41] J. Sproston, S. Donatelli, Backward stochastic bisimulation in CSL model checking, in: Proc. 1st Int. Conf. on the Quant. Eval. of Sys. (QEST'04), IEEE CS Press, 2004, pp. 220–229.
[42] E.W. Stark, On behavior equivalence for probabilistic I/O automata and its relation to probabilistic bisimulation, Journal of Automata, Languages and Combinatorics 8 (2) (2003) 361–396.
[43] W.J. Stewart, Introduction to the Numerical Solution of Markov Chains, Princeton University Press, 1994.
[44] R. van Glabbek, S. Smolka, B. Steffen, C. Tofts, Reactive, generative and stratified models for probabilistic processes, in: Proc. LICS'90, 1990, pp. 130–141.