Stochastic Reasoning About Channel-Based Component Connectors

Christel Baier[1] and Verena Wolf[2]

[1] Universität Bonn, Germany
baier@cs.uni-bonn.de
[2] Universität Mannheim, Germany
wolf@informatik.uni-mannheim.de

Abstract. Constraint automata have been used as an operational model for component connectors that coordinate the cooperation and communication of the components by means of a network of channels. In this paper, we introduce a variant of constraint automata (called continuous-time constraint automata) that allows us to specify time-dependent stochastic assumptions about the channel connections or the component interfaces, such as the arrival rates of communication requests, the average delay of enabled I/O-operations at the channel ends or the stochastic duration of internal computations. This yields the basis for a performance analysis of channel-based coordination mechanisms. We focus on compositional reasoning and discuss several bisimulation relations on continuous-time constraint automata. For this, we adapt notions of strong and weak bisimulation that have been introduced for similar stochastic models and introduce a new notion of weak bisimulation which abstracts away from invisible non-stochastic computations as well as the internal stochastic evolution.

1 Introduction

Coordination models and languages provide a formalization of the *glue-code* that binds individual components and organizes the communication and cooperation between them. In the past 15 years, various types of coordination models have been proposed that they yield a clear separation between the internal structure of the components and their interactions. See e.g. [19, 24, 13, 25, 15].

The purpose of this paper is to introduce an operational model for reasoning about *stochastic properties* of coordination languages similar to the approaches of Priami [27] and Di Pierro et al. [16]. In contrast to these approaches our focus is on exogenous channel-based coordination languages, such as Reo [2] (see also [5, 29, 1, 17, 14]) and stochastic models with nondeterminism. The rough idea of Reo is that complex component connectors are synthesized from channels via certain composition operators, thus yielding a *network of channels* (called Reo connector circuit) that coordinates the interactions between the components. An operational semantics of Reo connector circuits has been provided by means of *constraint automata* [4]. These are variants of labelled transition systems and encode the configurations of the network by their states and the possible data flow at the ports of the components and the nodes "inside" the network by their transitions. Extensions of constraint automata have been presented in [3] to study real-time constraints of component connectors and in [6] to reason about

P. Ciancarini and H. Wiklicky (Eds.): COORDINATION 2006, LNCS 4038, pp. 1–15, 2006.

channels with a probabilistic effect. The latter approach is time-abstract and deals with discrete probabilities, e.g., to model the faulty behaviours of buffered channels that might loose or corrupt stored messages, while the former approach focusses on a purely timed setting, e.g., to reason about hard deadlines, but does not deal with probabilities. The contribution of this paper is orthogonal to the extensions proposed in [3, 6] as it introduces a stochastic variant of constraint automata where transitions might have a certain delay according to some probability distribution over a continuous time domain. This model, called continuous-time constraint automata (CCA for short), combines the features of ordinary constraint automata with the race conditions in continuous-time Markov chains. CCA are close to *interactive Markov chains* (IMCs), which have been introduced by Hermanns [20] for specifying reactive systems with internal stochastic behaviours. CCA can be used – as ordinary constraint automata – to formalize the stepwise behaviour of the interfaces of the components and the channels connecting them, as well as an operational model for the composite system. In addition, CCA provide the possibility to specify, e.g., the average rate of communication requests of a certain component or the mean time that have to be passed between two consecutive I/O operations at a certain channel. Thus, CCA yield a simple and intuitive model that allow for a performance analysis of channel-based coordination mechanisms.

In this paper, we concentrate on compositional reasoning by means of *bisimulation* relations on CCA. We first consider strong and weak bisimulation, that have been introduced for interactive Markov chains [20]. These notions adapted to CCA yield equivalences that are congruences for the two basic operators (product and hiding) for generating the CCA for a complex component connector out of the CCA for its channels and the component interfaces. Furthermore, we introduce a new notion of weak bisimulation, called *very weak bisimulation* which abstracts away from the stochastic branching behaviour and cumulates the effect of sequences of stochastic transitions. Very weak bisimulation equivalence is coarser than weak bisimulation equivalence, but preserves the probabilities for trace-based linear time properties and can be checked in polynomial-time.

Organization. Section 2 recalls the basic concepts of ordinary constraint automata. CCA and a product and hiding operator on CCA are introduced in Section 3. Section 4 deals with bisimulation relations on CCA. Section 5 concludes the paper. Due to length restrictions, proofs for the theorems are omitted. They can be found in the full version (see `http://pi2.informatik.uni-mannheim.de/HomePages/vwolf/cca.ps`).

2 Constraint Automata

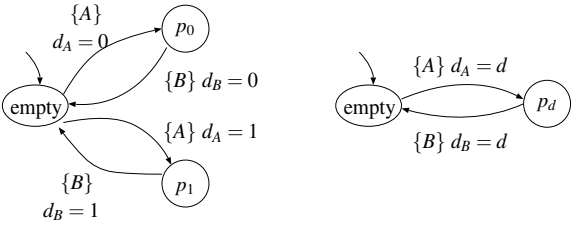
This section summarizes the basis concepts of constraint automata [4] and their use as operational model for channel-based component connectors. Constraint automata, CA for short, are variants of labelled transition systems where transitions are augmented with pairs $\langle N, g \rangle$ rather than action labels. The states of a constraint automata stand for the network configurations, e.g., the contents of the buffers for FIFO channels. The transition labels $\langle N, g \rangle$ can be viewed as *sets of I/O-operations* that will be performed *in parallel*. More precisely, N is a set of nodes in the network where data-flow is observed simultaneously, and g is a boolean condition on the observed data items. Thus,

transitions going out of a state q represent the possible data-flow in the corresponding configuration and its effect on the configuration.

Data assignments and data constraints. CA use a finite set $\mathcal{N}$ of nodes. The nodes can play the role of input and output ports of components, but they can appear outside the interfaces of components as "intermediate" stations of the network where several channels are glued together and the transmission of data items can be observed. For the purposes of this paper, there is no need to distinguish between write and read operations at the nodes. Instead CA only refer to the data items that can be "observed" at a node. Throughout the paper, we assume a fixed, non-empty and finite data domain *Data* consisting of the data items that can be transmitted through the channels. A data assignment for $N \subseteq \mathcal{N}$ means a function $\delta : N \to Data$. We write $\delta.A$ for the data item assigned to node $A \in N$ under δ and $DA(N)$ for the set of all data assignments for node-set N. CA use a symbolic representation of data assignments by data constraints which mean propositional formulae built from the atoms "$d_A = d_B$", "$d_A \in P$" or "$d_A = d$" where A, B are nodes, d_A is a symbol for the observed data item at node A and $d \in Data$, $P \subseteq Data$. The symbol $\models$ stands for the obvious satisfaction relation which results from interpreting data constraints over data assignments. Satisfiability and logical equivalence $\equiv$ of data constraints are defined as usual. We write $DC(N)$ to denote the set of satisfiable data constraints using only the symbols d_A for $A \in N$, but not d_B for $B \in \mathcal{N} \setminus N$.

Constraint automata (CA) [4]. A CA is a tuple $\mathcal{A} = (Q, \mathcal{N}, \longrightarrow, Q_0)$ where Q is a set of states, also called configurations, $\mathcal{N}$ a finite set of nodes, $Q_0 \subseteq Q$ the set of initial states and $\longrightarrow$ a subset of $\bigcup_{N \subseteq \mathcal{N}} Q \times \{N\} \times DC(N) \times Q$, called the transition relation.

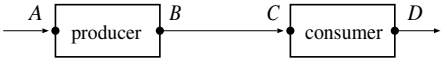
We write $q \xrightarrow{N,g} p$ instead of $(q, N, g, p) \in \longrightarrow$ and refer to N as the node-set and g the guard. Transitions where the node-set N is non-empty are called *visible*, while transitions with the empty node-set are called *hidden*. Each transition represents a set of possible interactions given by the *transition instances* that result by replacing the guard g with a data assignment δ where g holds. The intuitive behaviour of a CA is as follows. The automaton starts in an initial state. If the current state is q then an instance $q \xrightarrow{N,\delta} p$ of the outgoing transitions from q is chosen, the corresponding I/O-operations are performed and the next state is p. If there are several outgoing transitions from state q the next transition is chosen nondeterministically. A formalization of the possible (finite or infinite) observable data flow of a constraint automaton is obtained by the notion of a run. A run in $\mathcal{A}$ denotes a (finite or infinite) sequence of consecutive transition instances $q_0 \xrightarrow{N_0,\delta_0} q_1 \xrightarrow{N_1,\delta_1} q_2 \xrightarrow{N_2,\delta_2} \ldots$ where $q_0 \in Q_0$. For finite runs we require that the last state q does not have an outgoing hidden transition. This can be understood as a *maximal progress assumption* for hidden transitions, i.e., steps that do not require any interaction with the environment.



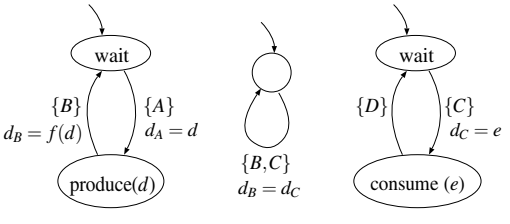
The picture above shows a CA for a FIFO1 channel with the node-set $\mathcal{N} = \{A, B\}$ and $Data = \{0, 1\}$. Node A serves as input port where data items can be written into the channel, while node B can be regarded as an output port where the stored data element is taken from the buffer and delivered to the environment.

State "empty" stands for the configuration where the buffer is empty, while state p_d encodes the configuration where d is stored in the buffer. Often, we use simplified parametric pictures for CA with meta symbols for data items as in the right of the picture (and formally explained in [4]).

For another example, we regard a simple system consisting of a producer and a consumer which are linked via a synchronous channel for transmitting the generated products from the producer's output port B to the consumer's input port C.



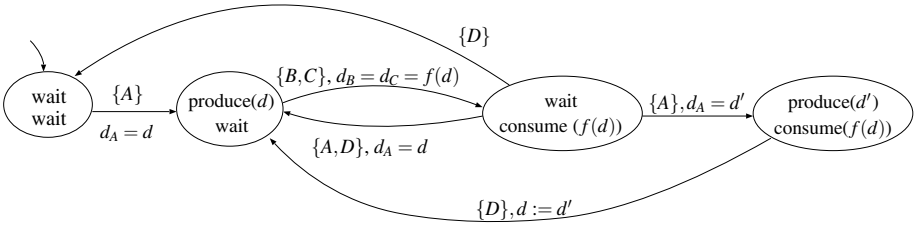
We model both components (producer and consumer) and the synchronous channel BC by CA. Parametric pictures are shown below. We assume here that the producer is activated by obtaining an input value d from the environment at its input port A. It then generates a certain product $f(d)$ which is synchronously delivered to the consumer. After having received $e = f(d)$, the consumer starts the consume-phase and finally sends a signal via output port D to the environment. We assume here that the value send off at D is arbitrary, that is, we deal with the valid guard `true` (which is omitted in the picture).



Product. To obtain a constraint automata for the composite producer-consumer-system, we apply a product construction to the three CA. The product of two CA $\mathcal{A}_1 = (Q_1, \mathcal{N}_1, \longrightarrow_1, Q_{0,1})$ and $\mathcal{A}_2 = (Q_2, \mathcal{N}_2, \longrightarrow_2, Q_{0,2})$ is defined as follows. $\mathcal{A}_1 \bowtie \mathcal{A}_2$ is a CA with the components $(Q_1 \times Q_2, \mathcal{N}_1 \cup \mathcal{N}_2, \longrightarrow, Q_{0,1} \times Q_{0,2})$ where $\longrightarrow$ is given by the following rules:

- If $q_1 \xrightarrow{N_1, g_1}_1 p_1$, $q_2 \xrightarrow{N_2, g_2}_2 p_2$, $N_1 \cap \mathcal{N}_2 = N_2 \cap \mathcal{N}_1 \neq \emptyset$ then $\langle q_1, q_2 \rangle \xrightarrow{N_1 \cup N_2, g_1 \wedge g_2} \langle p_1, p_2 \rangle$, provided that $g_1 \wedge g_2$ is satisfiable.
- If $q_1 \xrightarrow{N,g}_1 p_1$ where $N \cap \mathcal{N}_2 = \emptyset$ then $\langle q_1, q_2 \rangle \xrightarrow{N,g} \langle p_1, q_2 \rangle$.
- If $q_2 \xrightarrow{N,g}_2 p_2$ where $N \cap \mathcal{N}_1 = \emptyset$ then $\langle q_1, q_2 \rangle \xrightarrow{N,g} \langle q_1, p_2 \rangle$.

The former rule expresses the synchronization case which means that both automata have to "agree" on the I/O-operations at their common nodes, while the I/O-operations at their individual nodes is arbitrary. The latter two rules are in the style of classical interleaving rules for labelled transition systems. They formalize the case where no synchronization is required since no common nodes are involved. A parametric picture for the product of the CA of the producer, the consumer and the synchronous channel BC has the following form:



Hiding. Another operator that is helpful for abstraction purposes and can be used in Reo to built components from networks by declaring the internal structure of the network as hidden (i.e., invisible for the environment) is the hiding operator. It takes as input a CA $\mathcal{A} = (Q, \mathcal{N}, \longrightarrow, Q_0)$ and a non-empty node-set $M \subseteq \mathcal{N}$. The result is a CA $\mathsf{hide}(\mathcal{A}, M)$ that behaves as $\mathcal{A}$, except that data flow at the nodes $A \in M$ is made invisible. Formally, $\mathsf{hide}(\mathcal{A}, M) = \big(Q, \mathcal{N} \setminus M, \longrightarrow_M, Q_{0,M}\big)$ where

$$q \xrightarrow{\bar{N},\bar{g}}_M p \text{ iff there exists a transition } q \xrightarrow{N,g} p \text{ such that } \bar{N} = N \setminus M \text{ and } \bar{g} = \exists M[g].$$

$\exists M[g]$ stands short for $\bigvee_{\delta \in DA(M)} g[d_A/\delta.A \mid A \in M]$, where $g[d_A/\delta.A \mid A \in M]$ denotes the syntactic replacement of all occurrences of d_A in g for $A \in M$ with $\delta.A$. Thus, $\exists M[g]$ formalizes the set of data assignments for $\bar{N} = N \setminus M$ that are obtained from a data assignment δ for N where g holds by dropping the assignments for the nodes $A \in N \cap M$. For example, hiding nodes B and C in the CA $\mathcal{A}$ for the producer-consumer system yields a CA $\mathcal{A}' = \mathsf{hide}(\mathcal{A}, \{B,C\})$ with node-set $\{A,D\}$. $\mathcal{A}'$ has the same structure as $\mathcal{A}$, the only difference being that the $\{B,C\}$-transition in $\mathcal{A}$ becomes a hidden transition in $\mathcal{A}'$.

3 Continuous-Time Constraint Automata

We now present a stochastic extension of constraint automata that yields the basis for a performance analysis of channel-based component connectors, e.g. to reason about expected response times, the average number of messages that are stored in a buffer of a FIFO channel, the stochastic long-run behaviour or verifying soft deadlines such as "there is a 95% chance to obtain a message at input port B within 10 time units after having sent a request from output port A". Continuous-time constraint automata (CCA for short) rely on the assumption that hidden transitions are performed as soon as possible, while enabled I/O-operations at (non-hidden) nodes can occur at any moment or even can be refused. The idea is that the environment might connect to the non-hidden nodes and might either agree to perform a communication immediately, might cause a delay of a certain communication or might even be not willing to cooperate. CCA are most in the spirit of interactive Markov chains (IMC) that have been introduced by Hermanns [20] and that are closely related to continuous-time Markov decision processes [28]. As in IMCs we have two types of transitions:

- *interactive* transitions $q \xrightarrow{N,g} p$ as in ordinary constraint automata, and
- *Markovian transitions* $q \xrightarrow{\lambda} p$ where λ is a positive real number, called rate.

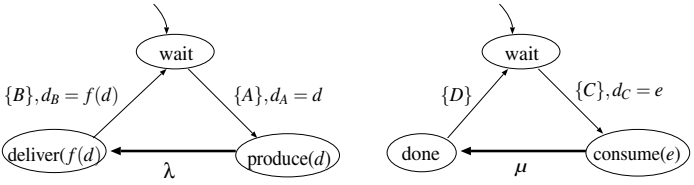
The interpretation of the rates is as in continuous-time Markov chains, see e.g. [22], i.e., with probability $1 - e^{-\lambda t}$ the delay of a Markovian transition with rate λ is less

or equal t. If there are two or more outgoing Markovian transitions from q and no interactive transition is taken from q then the transition with the least delay (i.e., the transition that is enabled first) will fire. Note that rates and average delays are dual in the sense that average delay Λ stands for the rate $\lambda = 1/\Lambda$.

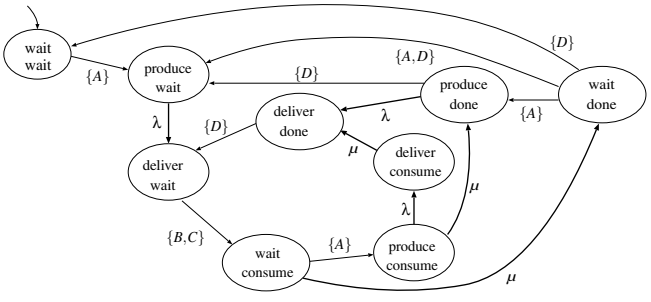
Definition 1 (Continuous-time constraint automata (CCA)). A CCA is a tuple $\mathcal{C} = (Q, \mathcal{N}, \longrightarrow, Q_0)$ where Q is a countable set of states, $Q_0 \subseteq Q$ the set of initial states, $\mathcal{N}$ is a finite set of nodes and $\longrightarrow \subseteq \ (Q \times \mathbb{R}_{>0} \times Q) \cup \left(\bigcup_{N \subseteq \mathcal{N}} Q \times \{N\} \times DC(N) \times Q\right)$ such that for all states q and p there is at most one Markovian transition from q to p. □

We write $\mathbf{R}(q,p) = \lambda$ if $q \xrightarrow{\lambda} p$ and $\mathbf{R}(q,p) = 0$ if there is no Markovian transition from q to p. For mathematical reasons, we require that for each state the exit rate $E(q)$ defined by $\sum_{p \in Q} \mathbf{R}(q,p)$ is finite and that there does not exist an infinite path consisting of consecutive interactive transitions. (The latter assumptions are irrelevant for the purposes of this paper, but they are necessary to ensure non-zenoness.) When state q is entered then either *immediately* a hidden transition instance is taken or the system stays in state q until one of the Markovian transitions becomes enabled and fires or a visible interactive transition is taken. A visible transition instance $q \xrightarrow{N,\delta} p$ can only be taken if all involved nodes $A \in N$ agree to perform the I/O-operations specified by (N, δ). If N is non-empty then this agreement depends on the (unknown) environment which might refuse to provide the required I/O-operations at the nodes $A \in N$. Thus, none of the visible transitions might be taken. If, however, the current state q has one or more outgoing hidden transitions there is a *nondeterministic choice* which selects one of the interactive (visible or hidden) transitions. Thus, Markovian transitions can only be taken from state q if there is no hidden transition that starts in q, in which case q is called a *Markovian state*.

The possible stepwise behaviours of a CCA can be made precise by means of the runs and the induced stochastic processes. A run in a CCA $\mathcal{C}$ is a sequence of consecutive transition instances $q_0 \xrightarrow{\alpha_0} q_1 \xrightarrow{\alpha_1} q_2 \xrightarrow{\alpha_2} \ldots$ where the α_i's are either triples (t, N, δ) such that $q_i \xrightarrow{N,\delta} q_{i+1}$ is an instance of an interactive transition and $t \geq 0$ (the time passage between entering state q_i and performing the I/O-operations specified by (N, δ)) or $\alpha_i \in \mathbb{R}_{>0}$ and there is a Markovian transition from q_i to q_{i+1}. In the latter case, α_i stands for the amount of time the system spends in state q_i until the first Markovian transition fires. According to the maximal progress assumption we require that Markovian transitions and that $\alpha_i = (t, N, \delta)$ for some $t > 0$ can only occur if no hidden transition can be taken in q_i and that finite runs end in a state where all outgoing transitions are visible. To reason about the probabilities of runs, the concept of schedulers, also often called policy, strategy or adversary, is needed. The details, which can be found e.g. in [28], are not of importance here. We just mention that a scheduler takes as input the history of the system, formalized by a finite prefix of a run, and either selects one of the enabled interactive transition instances or, if no hidden transition can be taken, decides to take a visible transition instance with some delay t unless a Markovian transition fires first or decides to take no interactive transition and to wait for the first enabled Markovian transition. For any given scheduler a probability measure on the induced runs can be defined which, for instance, allows to speak about the probability to reach a certain configuration within t time units or the expected time until a certain communication takes place.

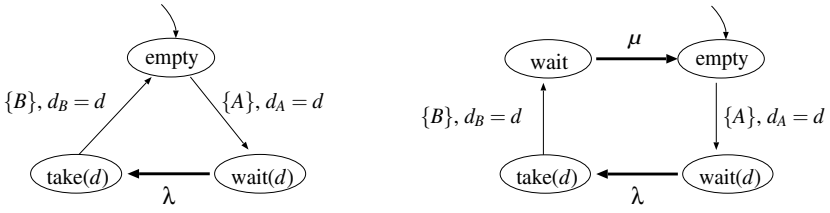


Example 1. The picture above shows a stochastic variant for the CA of the producer and the consumer. Here, we assume that the production time takes in average $1/\lambda$ time units, while the mean time of the consume phase is $1/\mu$. The data-abstract behavior of the composite system can then be specified by the CCA shown below. This CCA can now be subject of a stochastic analysis. For example, it can be verified that the average time of one production-consume cycle is $1/\lambda + 1/\mu$, or that the probability for the event "after being activated through an input at A, the time for delivering the product via channel BC is less or equal t" is given by $1 - e^{-\lambda t}$. □



Beside specifying stochastic phenomena that are internal to certain components, also channels might have stochastic behaviours and can be modelled by CCA. E.g., if a component Comp, that is linked to the sink end of a FIFO1 channel c, is waiting for a message along c then Comp cannot immediately read when a message is written at the source end. Instead it has to wait for a certain (possibly very small) amount of time until the read operation can be performed. As long as we consider any channel in isolation these delays might be very small or even negligible. However, for complex networks where several channels are composed, the effect of delays becomes less clear and can play a crucial role for performability issues. Assume a FIFO channel c with 1.000 buffer cells is composed from 1.000 copies of FIFO1 channel with average delay Λ then the mean time passage between writing a data item d into c's source end and the instant where d can be taken at the sink end is $1.000 \cdot \Lambda$.

Example 2. A FIFO1 channel with average delay $1/\lambda$ between the read and write operations can be modelled by one of the CCA shown below. In both CCA, after the write operation at the source A the state wait(d) is reached where a Markovian transition with rate λ is emanating, leading to state take(d) where the sink B can take the element. In the CCA on the left, no proper delay between the read operation at sink B and the next write operation at A is specified, while the automaton on the right relies on the assumption that the physical properties of the buffer yield an average delay $1/\mu$ for enabling a write operation after a read operation. □



We now explain how to construct a CCA for a given network, generated from channels and component interfaces with CCA-semantics, in a compositional manner. We assume here a calculus of channels, such as Reo [2], where networks are created via product (join) and hiding. In the following definition of the product of CCA we assume that the common nodes are those where data flow has to be synchronized.

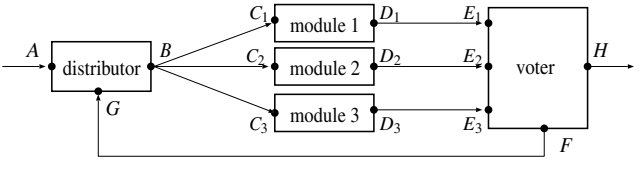
Definition 2 (Product of CCA). The product of two CCA $\mathcal{C}_1 = (Q_1, \mathcal{N}_1, \longrightarrow_1, Q_{0,1})$ and $\mathcal{C}_2 = (Q_2, \mathcal{N}_2, \longrightarrow_2, Q_{0,2})$ is the CCA $\mathcal{C}_1 \bowtie \mathcal{C}_2 = (Q_1 \times Q_2, \mathcal{N}_1 \cup \mathcal{N}_2, \longrightarrow, Q_{0,1} \times Q_{0,2})$ where $\longrightarrow$ is defined by the synchronization and interleaving rule for interactive transitions as in ordinary CA (see Section 2) and the following interleaving rules for the Markovian transitions:

$$\frac{q_1 \xrightarrow{\lambda}_1 p_1}{\langle q_1, q_2 \rangle \xrightarrow{\lambda} \langle p_1, q_2 \rangle} \qquad \frac{q_2 \xrightarrow{\lambda}_2 p_2}{\langle q_1, q_2 \rangle \xrightarrow{\lambda} \langle q_1, p_2 \rangle}$$

The interleaving rule for the Markovian transition is adequate due to the memory-less property of exponential distributions. The resulting interleaving diamond for a state $\langle q_1, q_2 \rangle$ models the "race" of the Markovian transitions in q_1 and those in q_2.

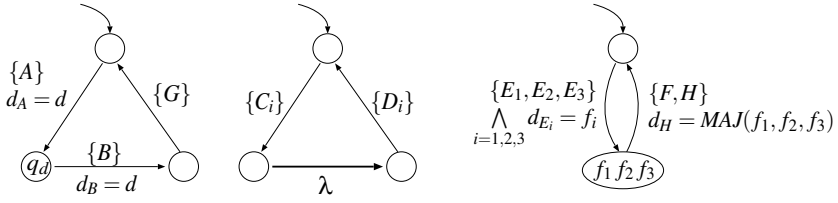
The product-operator $\bowtie$ is associative and commutative (up to isomorphism). Thus, when starting with a network where several components are linked via channels then the CCA for the composite system is obtained by applying the binary operator $\bowtie$ to the CCA for the channels and component interfaces in any order.

Example 3 (Triple modular redundancy). Let us look for a CCA that models a fault tolerant system relying on von Neumann's concept of triple modular redundancy. The task is to compute a certain boolean function value $f(d)$ for an input value $d \in \{0,1\}$ provided by a user and to return $f(d)$. Three unreliable modules are available that attempt to calculate $f(d)$, but may fail to compute the correct value. Thus, after having obtained the computed values $f_1, f_2, f_3 \in \{0,1\}$ by the modules, a majority decision will be made and the value $MAJ(f_1, f_2, f_3)$ will be returned to the user.

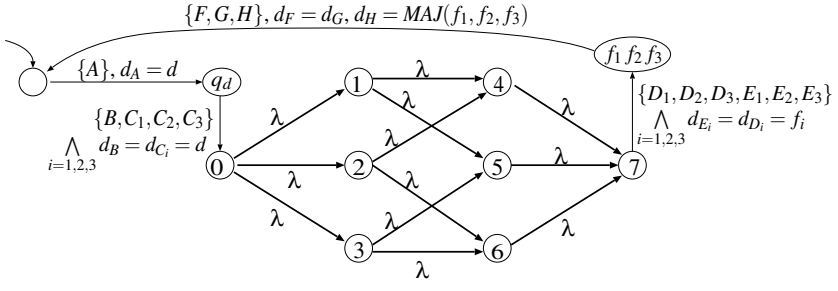


The system consists of five components as shown above. The distributor gets the input value d via its input port A and delivers it to the three modules via synchronous

channels connecting the distributor's output port B with the input ports C_i of the modules. The modules operate independently from each other and calculate a values f_i that will be delivered via a synchronous channel D_iE_i. The voter then makes the majority decision and returns the obtained value via its output port H. To avoid that the distributor reads the next input value before the voter has returned a value, the voter and the distributor are linked via a synchronous channel FG. Assuming that the average time for the internal computation of the modules is $1/\lambda$ and all other transitions are immediate the interfaces of the distributor, modules and voter can be modelled by the following CCA:



Composing these CCA with the automata for the involved synchronous channels via the product-operator $\bowtie$ yields the CCA shown below. □



Definition 3 (Hiding in CCA). Let $\mathcal{C} = (Q, \mathcal{N}, \longrightarrow, Q_0)$ be a CCA and $\emptyset \neq M \subseteq \mathcal{N}$. The CCA $\mathsf{hide}(\mathcal{C}, M)$ is the tuple $\big(Q, \mathcal{N} \setminus M, \longrightarrow_M, Q_{0,M}\big)$ where the transition relation $\longrightarrow_M$ is given by the following two rules:

$$\frac{q \xrightarrow{N,g} p,\ \ \bar{N} = N \setminus M,\ \ \bar{g} = \exists M[g]}{q \xrightarrow{\bar{N},\bar{g}}_M p} \quad \text{and} \quad \frac{q \xrightarrow{\lambda} p,\ \lambda > 0}{q \xrightarrow{\lambda}_M p}$$

where $\exists M[g]$ is defined as in the non-probabilistic case. □

4 Bisimulation on CCA

Since CCA are slight variants of interactive Markov chains (IMCs), we may adapt the bisimulation techniques suggested in [20] for IMCs. Bisimulation equivalence for IMCs arises through a combination of standard bisimulation [23] for the interactive transitions and lumping equivalence [12, 21, 9, 8] for the Markovian transitions. We now adapt

these notions of strong and weak bisimulation for CCA and introduce a new (coarser) variant of weak bisimulation equivalence on CCA.

For $P \subseteq Q$, $\mathbf{R}(q,P) = \sum_{p \in P} \mathbf{R}(q,p)$ denotes the total rate to move from q to P via Markovian transitions.

Definition 4 (Strong bisimulation for CCA). Let $\mathcal{C}$ be a CCA as in Def. 1. A strong bisimulation on $\mathcal{C}$ is an equivalence $\mathcal{R}$ on Q such that for all $(q_1,q_2) \in \mathcal{R}$:

(S1) If $q_1 \xrightarrow{N,\delta} p_1$ then there is a transition instance $q_2 \xrightarrow{N,\delta} p_2$ such that $(p_1,p_2) \in \mathcal{R}$.
(S2) If there is no outgoing hidden transition from q_1 then $\mathbf{R}(q_1,P) = \mathbf{R}(q_2,P)$ for all equivalence classes $P \in Q/\mathcal{R}$.

Two states q_1, q_2 are called strongly bisimilar in $\mathcal{C}$, denoted $q_1 \sim_{\mathcal{C}} q_2$ (or briefly $q_1 \sim q_2$), if the pair (q_1,q_2) is contained in some strong bisimulation. Strong bisimulation equivalence for two CCA $\mathcal{C}_1$ and $\mathcal{C}_2$ with the same node-set is defined by considering the CCA $\mathcal{C} = \mathcal{C}_1 \uplus \mathcal{C}_2$ that results from the disjoint union of $\mathcal{C}_1$ and $\mathcal{C}_2$ and requiring that for each initial state q_1 in $\mathcal{C}_1$ there exists an initial state $q_2 \in \mathcal{C}_2$ such that $q_1 \sim_{\mathcal{C}} q_2$, and vice versa. We write $\mathcal{C}_1 \sim \mathcal{C}_2$ to denote that $\mathcal{C}_1$ and $\mathcal{C}_2$ are strongly bisimilar. □

Condition (S2) makes no restrictions on the Markovian transitions if q_1 (and hence also q_2 by (S1)) has an interactive transition with the empty node-set. In fact, any state that has such an internal move (which represents data flow at some hidden nodes) will immediately be left by one of the enabled interactive transitions. Thus, the Markovian transitions are irrelevant for them, and could simply be removed.

In an analogous way, we adapt Hermans's notion of weak bisimulation for IMCs [20] to our setting. CCA $\mathcal{C}$ is said to have a weak hidden transition from q to p, denoted $q \Longrightarrow p$, if p is reachable from q in $\mathcal{C}$ via zero or more hidden transitions. Furthermore,

$$q \xRightarrow{N,g} p \text{ iff there exists states } q' \text{ and } p' \text{ with } q \Longrightarrow q',\ q' \xrightarrow{N,g} p' \text{ and } p' \Longrightarrow p.$$

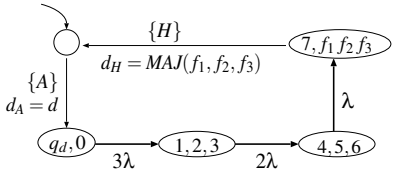
We refer to $q \xRightarrow{N,\delta} p$ as a weak interactive transition instance in $\mathcal{C}$. For $P \subseteq Q$, the backward closure of P in $\mathcal{C}$, denoted $bc(P)$, is the set of all states $q \in Q$ such that $q \Longrightarrow p$ for some state $p \in P$. Thus, $bc(P)$ contains exactly those states that can reach a P-state via hidden transitions. Since hidden transitions are always enabled, the states $q \in bc(P)$ are those that can reach P without any delay. State q is called Markovian if there is no outgoing hidden transition from q.

Definition 5 (Weak bisimulation for CCA). Let $\mathcal{C}$ be a CCA as in Def. 1. A weak bisimulation on $\mathcal{C}$ is an equivalence $\mathcal{R}$ on Q such that for all $(q_1,q_2) \in \mathcal{R}$:

(W1) Each weak interactive transition instance of q_1 can be matched by a weak interactive transition instance of q_2 in the following sense:
(W1.1) If $q_1 \xRightarrow{N,\delta} p_1$ where $N \neq \emptyset$ then $q_2 \xRightarrow{N,\delta} p_2$ for some state p_2 where $(p_1,p_2) \in \mathcal{R}$.
(W1.2) If $q_1 \Longrightarrow p_1$ then $q_2 \Longrightarrow p_2$ for some state p_2 such that $(p_1,p_2) \in \mathcal{R}$.
(W2) If q_1 is Markovian then there exists a Markovian state r_2 with $q_2 \Longrightarrow r_2$, $(q_1,r_2) \in \mathcal{R}$ and $\mathbf{R}(q_1,bc(P)) = \mathbf{R}(r_2,bc(P))$ for all $\mathcal{R}$-equivalence classes P.

Two states q_1, q_2 are called weakly bisimilar in $\mathcal{C}$, denoted $q_1 \approx_{\mathcal{C}} q_2$ (or briefly $q_1 \approx q_2$), if the pair (q_1, q_2) is contained in some weak bisimulation. Weak bisimilarity for two CCA is defined as in the case of strong bisimulation equivalence. □

Example 4 (Triple modular redundancy). Let us look for the product-CCA for TMR in Example 3. States 1,2,3 are weakly bisimilar, as well as states $4,5,6$. This simply follows by the fact that none of these states has a weak interactive transition and that $\mathbf{R}(i,\{4,5,6\}) = 2\lambda$ for $i = 1,2,3$ and $\mathbf{R}(i,\{7\}) = \lambda$ for $i = 4,5,6$. After hiding all nodes except for A and H, also states states q_d and 0 as well as states 7 and $\langle f_1 f_2 f_3 \rangle$ are weakly bisimilar. Note that e.g. for q_d and 0, conditions (W1.1) and (W1.2) are obviously fulfilled since none of these states has a weak visible transition and $q_d \Longrightarrow 0$.



To mimic the Markovian transitions of state 0, state q_d may first take the hidden transition from q_d to 0 and then perform the CTMC-like race in state 0. By collapsing weakly bisimilar states, we obtain a CCA with four states (see the picture on the left) that is weakly bisimilar to the original one. □

Bravetti [11] introduced a slightly coarser notion of weak bisimulation on IMCs, which relaxes condition (W2) by the requirement that $\mathbf{R}(q_1, bc(P)) = \mathbf{R}(r_2, bc(P))$ for all $\mathcal{R}$-equivalence classes P, except for the $\mathcal{R}$-equivalence class of q_1. Bravetti's weak bisimulation, denoted $\approx^B$, agrees with the coarsest equivalence satisfying condition (W1) and the following condition (W2'):

(W2') If q_1 is Markovian then $q_2 \Longrightarrow r_2$ for some Markovian state r_2 with $(q_1, r_2) \in \mathcal{R}$ and $\Pr(q_1 \stackrel{\leq t}{\Longrightarrow} bc(P)) = \Pr(r_2 \stackrel{\leq t}{\Longrightarrow} bc(P))$ for all $t \geq 0$ and $P \in Q/\mathcal{R}$ with $q_1 \notin P$.

where, for $t \geq 0$, $q \in Q$ and $P \subseteq Q$, $\Pr(q \stackrel{\leq t}{\Longrightarrow} P)$ denotes the probability to move from q to P via Markovian transitions emanating from Markovian states within at most t time units.[1]

Theorem 1 (Compositionality of strong and weak bisimulation). *Let $\mathcal{C}_1$, $\mathcal{C}_1'$, $\mathcal{C}_2$, $\mathcal{C}_2'$ be CCA such that $\mathcal{C}_i$ and $\mathcal{C}_i'$ have the same node-set, $i = 1,2$. Moreover, let $\cong$ be one of the three equivalences $\sim$, $\approx$ or $\approx^B$. Then, $\mathcal{C}_1 \cong \mathcal{C}_1'$ and $\mathcal{C}_2 \cong \mathcal{C}_2'$ implies $\mathcal{C}_1 \bowtie \mathcal{C}_2 \cong \mathcal{C}_1' \bowtie \mathcal{C}_2'$ and $\mathsf{hide}(\mathcal{C}_1, M) \cong \mathsf{hide}(\mathcal{C}_1', M)$.*

The congruence result stated in Theorem 1 yields that strong and weak bisimulation are adequate for the compositional design of complex component connectors. However, for the analysis coarser equivalences that abstract away from sequences of non-observable (hidden or Markovian) transitions, but still preserve the probabilities of the observable data flow are desirable.

We now present a new notion of bisimulation equivalence for CCA, called *very weak bisimulation*, which relies on the assumption that the given CCA models the "complete"

[1] The Markovian states together with their Markovian transitions yield an ordinary continuous-time Markov chain with state space Q. Thus, we may deal here with the standard sigma-algebra and probability measure on CTMCs, see e.g. [22, 26].

closed system where the enabledness of visible transitions no longer depend on the environment and can be taken as soon as possible. This view is adequate, even for systems that are open in nature provided that stochastic assumptions are available for the environment which allow to model the environment by a CCA. The CCA that is subject for the (stochastic) analysis then arises through the product of the CCA for the channels, the component interfaces and the environment.

In the sequel, let $\mathcal{C}$ be a CCA as in Def. 1. State q is called *purely Markovian* if q is Markovian and has no (hidden or visible) outgoing interactive transition. State q is called *vanishing* if q has exactly one hidden transition and no visible transitions. Let $\mathcal{C}^M$ be the CTMC that results from $\mathcal{C}$ after the following two transformations: 1) First, all vanishing states are replaced by their (unique) successor which is not vanishing, i.e. either purely Markovian, has more than one hidden transitions or has at least one visible transition. Note that the successor is not necessarily a direct one since a maximal sequence of vanishing states is replaced by the direct successor of the last state of this sequence. After this transformation there might still be states with hidden transitions (namely those that have at least one interactive transition in addition or two or more hidden transitions). 2) Next, all states that have at least one outgoing interactive transition are made absorbing, i.e. we remove all outgoing Markovian transitions of states that are not purely Markovian.

Then, for a purely Markovian state $q \in Q$, $P \subseteq Q$ and $t \geq 0$, $\Pr^M(q \stackrel{\leq t}{\Longrightarrow} P)$ denotes the probability to move from q to a state $p \in P$ in the CTMC $\mathcal{C}^M$ within at most t time units.

Definition 6 (Very weak bisimulation). A very weak bisimulation on a CCA $\mathcal{C}$ is an equivalence relation $\mathcal{R}$ on Q such that for all $(q_1, q_2) \in \mathcal{R}$:

(V1) Each weak interactive transition instance of q_1 can be matched by a weak interactive transition instance of q_2 in the sense of (W1.1) and (W1.2) of Definition 5.

(V2) If q_1 is purely Markovian then there exists a purely Markovian state r_2 with $q_2 \Longrightarrow r_2$, $(q_1, r_2) \in \mathcal{R}$ and

$$\Pr^M(q_1 \stackrel{\leq t}{\Longrightarrow} bc(P)) = \Pr^M(r_2 \stackrel{\leq t}{\Longrightarrow} bc(P))$$

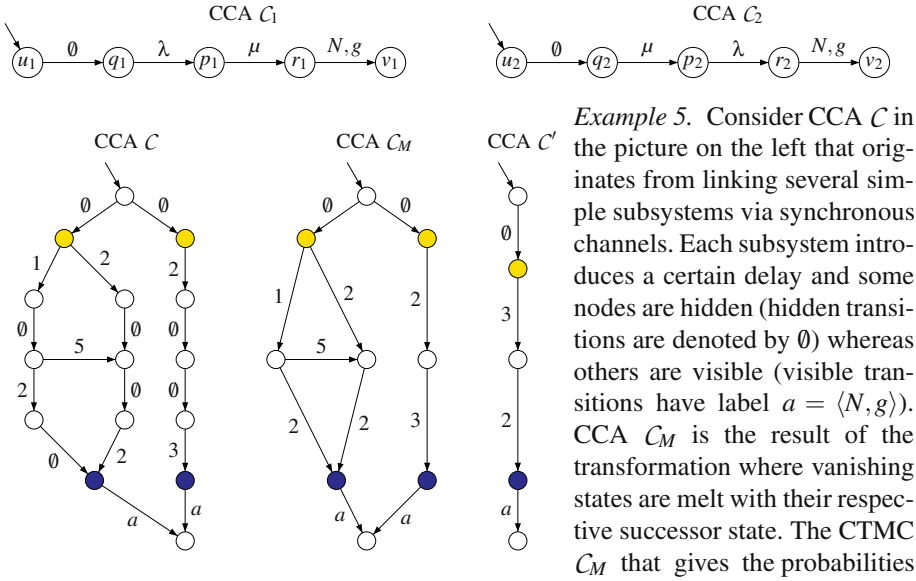
for all $\mathcal{R}$-equivalence classes P that contain no purely Markovian states.

States q_1, q_2 are called very weakly bisimilar in $\mathcal{C}$, denoted $q_1 \approx_{\mathcal{C}}^{vw} q_2$ (or briefly $q_1 \approx^{vw} q_2$), if the pair (q_1, q_2) is contained in $\mathcal{R}$ for some very weak bisimulation $\mathcal{R}$. Very weak bisimilarity for two CCA is defined as for strong bisimulation equivalence. □

The rationale behind (V2) is that $\approx^{vw}$-equivalent states can reach the set $bc(P)$ after some time passage t (obtained by sequences of Markovian and hidden transitions leading to a state p that is in the backward closure of P) with equal probability.

Note that there are no restrictions on Markovian transitions of states that are not purely Markovian. The relation $\approx^{vw}$ abstracts away from those transitions which is a useful abstraction for closed models where visible transitions will never be delayed due to the product operator and can be considered as immediate. Clearly, $\approx$ is strictly finer than $\approx^B$ which again is finer than $\approx^{vw}$ since for each weak bisimulation $\mathcal{R}$ à la Bravetti $\mathcal{R}$ is also a very weak bisimulation.

For the CCA $\mathcal{C}_1, \mathcal{C}_2$ in the following picture we have $\mathcal{C}_1 \approx^{vw} \mathcal{C}_2$, while $\mathcal{C}_1 \not\approx^{B} \mathcal{C}_2$, provided that $\lambda, \mu > 0$, $\lambda \neq \mu$ and $N \neq \emptyset$.



Example 5. Consider CCA $\mathcal{C}$ in the picture on the left that originates from linking several simple subsystems via synchronous channels. Each subsystem introduces a certain delay and some nodes are hidden (hidden transitions are denoted by $\emptyset$) whereas others are visible (visible transitions have label $a = \langle N, g \rangle$). CCA $\mathcal{C}_M$ is the result of the transformation where vanishing states are melt with their respective successor state. The CTMC $\mathcal{C}_M$ that gives the probabilities for condition (V2) in Definition 6 can be obtained by considering only the Markovian transitions of CCA $\mathcal{C}_M$. It holds that $\mathcal{C} \approx^{vw} \mathcal{C}_M \approx^{vw} \mathcal{C}'$ because the probability to reach a dark shaded state within $t > 0$ time units from a light shaded state is equal in all three depicted CCA. This comes from the fact that the underlying CTMCs have the same distribution when considering the time until an absorbing state (the dark shaded state) is entered. All initial states are in the same equivalence class, all light shaded states form an equivalence class, all dark shaded states form another class and the states reached via the visible a-transition are in the same class. All remaining states form singletons. □

For automatic reasoning with CCA in bisimulation framework, one aims at efficient algorithms for checking the equivalence of two finite CCA. Algorithms for checking strong and weak bisimulation equivalence have been proposed by Hermanns [20] for IMCs and can easily be adapted for CCA. These algorithms operate with a partition-splitter technique for computing the bisimulation quotient.

The treatment of very weak bisimulation is more difficult since it requires reasoning about *phase-type distributions* rather than (rates of) exponential distributions. The algorithm for checking $\approx^{vw}$ also relies on a partitioning splitter technique as it is standard for other bisimulation relations. In the sequel, we concentrate only on the rough ideas of a polynomial time algorithm. Several optimizations are possible to obtain a more efficient implementation.

First, vanishing states are replaced by their successor. Let n be the number of all involved states. Then as a preprocessing step for each pair (q, r) of states where q is purely Markovian and r not, the first n moments of the distribution of the time until absorption in r when q is the initial state are computed. This can be done in polynomial time. Since these distributions are uniquely determined by their first n moments during the partition-splitter algorithm the distributions for the time until the backward closure

of a certain equivalence class is reached can be checked for equality in an efficient way. For details we refer to the full version which can be found at our website (see `http://pi2.informatik.uni-mannheim.de/HomePages/vwolf/cca.ps`).

Theorem 2 (Equivalence checking). *Given two finite CCA $\mathcal{C}_1$ and $\mathcal{C}_2$ with the same node-set and over fixed data domain, the equivalence checking problem asking whether "Does $\mathcal{C}_1 \cong \mathcal{C}_2$ hold?" where $\cong$ is $\sim$, $\approx$, $\approx^B$ or $\approx^{vw}$ can be decided in polynomial time with respect to the total number of transitions and the total number of states in $\mathcal{C}_1$ and $\mathcal{C}_2$.*

We suggest to use Bravetti's notion of weak bisimulation for open models, i.e. models containing ports that require interaction with the environment. For closed models the coarser relation $\approx^{vw}$ gives a more abstract view on the model and still preserves linear-time properties [30]. Unlike weak bisimulation equivalence $\approx$ or $\approx^B$ very weak bisimulation is not a congruence for the product operator. This, however, is not surprising since its definition relies on the view of the given CCA as a closed model.

5 Conclusion

The goal of the paper was to provide an operational model for reasoning about component connectors under stochastic assumptions about the channels and component interfaces. We introduced CCA for this purpose, together with notions of strong and weak bisimulation that are preserved by the composition operators product and hiding. Since the latter are the only operators needed for the compositional generation of (static) networks in the channel-based calculus Reo, our framework fits well in this context and provides the basis for a performance analysis of Reo component connectors. In this paper, we concentrated on the issue of bisimulation relations for CCA. However, since CCA are close to standard stochastic models (such as continuous-time Markov decision processes), also other validation techniques are applicable, such as simulation on the basis of MoDeST [10] or reasoning about expectations [18] or verifying time-bounded reachability properties [7]. Vice versa, our new notion of weak bisimulation equivalence $\approx^{vw}$ can also be helpful for reasoning about IMCs and similar stochastic models.

References

1. F. Arbab. Abstract behavior types: A foundation model for components and their composition. In *[15]*, pages 33–70, 2003.
2. F. Arbab. Reo: A channel-based coordination model for component composition. *Mathematical Structures in Computer Science*, 14(3):1–38, 2004.
3. F. Arbab, C. Baier, F. de Boer, and J. Rutten. Models and temporal logics for timed component connectors. In *Proc. SEFM'04*. IEEE CS Press, 2004.
4. F. Arbab, C. Baier, J.J.M.M. Rutten, and M. Sirjani. Modeling component connectors in reo by constraint automata. *Science of Computer Programming,* special issue on Foundations of Coordination Languages and Software Architectures (to appear), 2005.
5. F. Arbab and J.J.M.M. Rutten. A coinductive calculus of component connectors. In *Proc. WADT 2002*, volume 2755 of *LNCS*, pages 35–56, 2003.
6. C. Baier. Probabilistic models for reo connector circuits. *Journal of Universal Computer Science*, 11(10):1718–1748, 2005.

7. C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Efficient computation of time-bounded reachability probabilities in uniform continuous-time markov decision processes. In *Proc. TACAS*, volume 2988 of *Lecture Notes in Computer Science*, pages 61–76, 2004. Full version to appear in Theoretical Computer Science.
8. C. Baier, H. Hermanns, J.-P. Katoen, and V. Wolf. Comparative branching time semantics for Markov chains. In *Proc. CONCUR 2003*, number 2761 in LNCS, pages 492–507, 2003. Full version to appear in Information and Computation.
9. M. Bernardo and R. Gorrieri. Extended Markovian process algebra. In *Proc. CONCUR 1996*, number 1119 in LNCS, pages 315–330. Springer, 1996.
10. H. Bohnenkamp, H. Hermanns, J.-P. Katoen, and R. Klaren. The modest modeling tool and its implementation. *Computer Performance Evaluation/TOOLS*, pages 116–133, 2003.
11. M. Bravetti. Revisiting interactive Markov chains. In *Proc. Models for Time-Critical Systems*, volume 68(5) of *Electr. Notes Theor. Comput. Sci.*, 2003.
12. P. Buchholz. Exact and ordinary lumpability in finite markov chains. *Journal of Applied Probability*, 31:59–75, 1994.
13. P. Ciancarini. Coordination models and languages as software integrators. *ACM Comput. Surv.*, 28(2):300–302, 1996.
14. D. Clarke, D. Costa, and F. Arbab. Modeling coordination in biological systems. In *Proc. of the Int. Symposium on Leveraging Applications of Formal Methods (ISoLA 2004)*, 2004.
15. F.S. de Boer, M.M. Bonsangue, S. Graf, and W.-P. de Roever, editors. *Formal Methods for Components and Objects*, volume 2852 of *LNCS*. Springer, 2003.
16. A. Di Pierro, C. Hankin, and H. Wiklicky. Continuous-time probabilistic klaim. *Electr. Notes Theor. Comput. Sci.*, 128(5):27–38, 2005.
17. N. Diakov and F. Arbab. Compositional construction of web services using Reo. In *Proc. International Workshop on Web Services: Modeling, Architecture and Infrastructure (ICEIS 2004), Porto, Portugal, April 13-14*, 2004.
18. E. Feinberg. Continuous time discounted jump markov decision processes: A discrete-event approach. *Math. Oper. Res.*, 29(3):492–524, 2004.
19. D. Gelernter and N. Carriero. Coordination languages and their significance. *Commun. ACM*, 35(2):97–107, 1992.
20. H. Hermanns. *Interactive Markov Chains*, volume 2428 of *Lecture Notes in Computer Science*. Springer Verlag, 2002.
21. J. Hillston. *A compositional approach to performance modelling*. Cambridge University Press, 1996.
22. J. G. Kemeny, J. L. Snell, and A. W. Knapp. *Denumerable Markov Chains*. D. Van Nostrand Co., Princeton, NJ, USA, 1966.
23. R. Milner. *Communication and Concurrency*. Prentice Hall International Series in Computer Science. Prentice Hall, 1989.
24. O. Nierstrasz, S. Gibbs, and D. Tsichritzis. Component-oriented software development. *Commun. ACM*, 35(9):160–165, 1992.
25. A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, editors. *Coordination of Internet Agents: Models, Technologies, and Applications*. Springer, 2001.
26. P. Panangaden. Measure and probability for concurrency theorists. *Theoretical Computer Science*, 253(2):287–309, 2001.
27. Corrado Priami. Stochastic pi-calculus. *Comput. J.*, 38(7):578–589, 1995.
28. M. L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, 1994.
29. J.J.M.M. Rutten. Component connectors. In *[?]*, chapter 5, pages 73–87. Oxford University Press, 2004.
30. V. Wolf, C. Baier, and M. Majster-Cederbaum. Trace semantics for stochastic systems with nondeterminism. In *Proc. QAPL*, 2006. to appear.