$\mathsf{IMC}_{\texttt{Reo}}$: interactive Markov chains for `Stochastic Reo`

Nuno Oliveira
HASLab INESC TEC
Universidade do Minho
nuno.s.oliveira@insectec.pt

Alexandra Silva
Centrum Wiskunde & Informatica
Radboud University Nijmegen
alexandra@cs.ru.nl

Luís S. Barbosa
HASLab INESC TEC
Universidade do Minho
luis.s.barbosa@insectec.pt

Abstract

The study of composed software and its properties is in itself a great challenge. Bringing quantitative aspects into the picture increases the complexity of the analysis and entails the need for reconciling both the classical and the stochastic analyses. The quest herein is to provide constructs for composition of building blocks and semantic models thereof which enable both analyses and allow for modeling and verification.

`Stochastic Reo` offers constructs for component and service coordination and provides means for specification of stochastic values for software connectors. Interactive Markov chains (`IMC`) on the other hand, are a stochastic, compositional, extension of classical models of concurrency, integrating classical and stochastic aspects.

This paper discusses how `IMC` can be effectively used as a compositional semantic model for `Stochastic Reo`, avoiding to resort to a separate automaton model, often lacking full compositionality, as intermediate semantics. A tool chain integrating available tool support for `IMC`, is described and applied in the analysis of a case study dealing with performance and resource allocation of a mediation system.

1 Introduction

The development of complex, software intensive systems within the service-oriented paradigm proceeds by interconnecting independent *loci* of computation, often running in different physical locations and provided by different organisations, according to specific protocols which constraint their mutual interaction to achieve a common goal. Such a *glue code* layer is typically abstracted into a coordination model which plays a major role in the specification and analysis of composed software.

`Reo` [2] is one such model in which exogenous coordination of autonomous services/components is achieved through different types of software connectors built themselves in a compositional way from a set of primitive channels. Its stochastic extension — known as `Stochastic Reo` [36] — allows for the annotation of channels with stochastic values that model the rates of arrival of requests to their borders and the corresponding processing delays.

Introducing stochastic behaviour in a coordination model makes possible to reason rigorously about quality of service (QoS) aspects of composed software. Quantitative analysis grew increasingly popular as service-orientation became the *de facto* paradigm for highly-dependable software systems, where performance and resource usage emerge as essential non-functional requirements. If the study of composed software and its properties is in itself a great challenge, with QoS aspects also in play the complexity of the analysis increases and it is a challenge to reconcile both the classical and the stochastic analyses.The quest herein is to provide constructs for composition of building blocks and semantic models thereof which enable both analyses and allow for modelling and verification.

Interactive Markov chains (`IMCs`) were proposed in [24] as a stochastic, compositional, extension of classical models of concurrency, in which the classical and stochastic features are treated in an harmonious manner. Furthermore, the model has extensive tool support for analysis and model checking, CADP [21] and IMCA [22] being popular examples. It is therefore reasonable to discuss to what extent `IMCs` can be effectively used to serve as semantic model for `Stochastic Reo`.

The paper extends our previous work along this path, documented in [39]. The basic idea is that treating IMCs as a first-call citizen model, instead of using a separate automaton model as intermediate semantics, gives rise to more faithful models and has the obvious efficiency advantages. Moreover, and crucial in reasoning about composed systems, it avoids the lack of full compositionality present in some automata models. Finally, this enables, without significant effort, the use of typical IMC tools and associated techniques to reason about qualitative and quantitative aspects of Stochastic Reo models.

Contributions. This work provides a detailed account of this research programme, extending our ACM SAC'2014 (SOAP track) paper [39] as follows:

- results concerning compositionality are extended to behavioural equivalence and substitutability,
- all proofs are included and extra details developed within the examples,
- tool support, as well as more detailed techniques of analysis, are discussed,
- the formal framework is applied to a new case study.

Outline. The paper is organised as follows. Section 2 recalls the basic definitions of IMC and stochastic Reo. The IMC$_{\text{Reo}}$ framework is introduced in detail in Section 3. Section 4 characterises composition operators for IMC$_{\text{Reo}}$ and investigates their properties. A tool chain for analysis of Stochastic Reo connectors in this setting, along with a description of its application to (a fragment of) a case study is presented in Section 5. Finally, related work and future research directions are discussed in Sections 6 and 7, respectively.

2 Background

This section recalls, for future reference, basic definitions and results on interactive Markov chains (IMC), the coordination language Reo, and its stochastic variant.

2.1 Reo and Stochastic Reo

Reo [5, 1, 2] is a channel-based model for exogenous coordination. A channel is a normally directed, communication device with exactly two ends: a source and a sink end; but Reo also accepts undirected channels (*i.e.*, channels with two ends of the same sort). A channel is synchronous when it delays the operations at each of its ends so that they can succeed simultaneously. Otherwise it is classified as asynchronous, exhibiting memory capabilities or the possibility of specifying an ordering policy for content delivering. Moreover, it may also be lossy when it delivers some values but loses others depending on a specified policy. Figure 1 depicts the basic channels used in Reo.

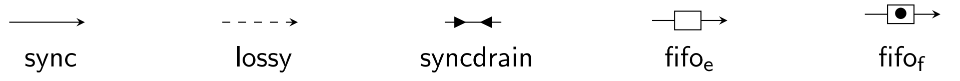


Figure 1: Primitive Reo channels.

The sync channel transmits data from one end to another whenever a request is simultaneously present at both of them, otherwise one request has to wait for the other. The lossy channel behaves likewise, but data may be lost whenever a request at the source end is not matched by another one at the sink end. Differently, a fifo channel has buffering capacity of (usually) one memory position, therefore allowing for asynchronous occurrence of input/output (I/O) requests. The qualifiers e or f refer to the channel internal state (either *empty* or *full*). Finally, the syncdrain channel accepts data synchronously at both ends and loses it.

Channels are combined by plugging their ends to define more complex coordination structures — all of them referred to as *software connectors*. A node may be of three distinct types: (*i*) source node, if it connects only source channel ends, (*ii*) sink node, if it connects only sink channel ends and (*iii*) mixed node, if it connects both. Source and sink nodes constitute the connector ports. Three typical connectors are depicted in Figure 2.

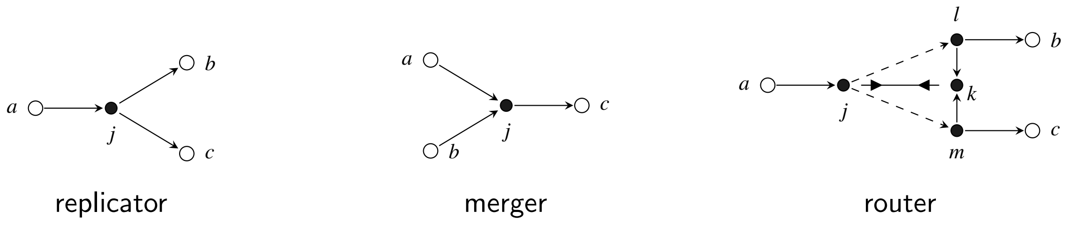


Figure 2: Some Reo connectors.

The replicator copies data flowing from port a to ports b and c, in parallel, through the mixed node j. This behaviour, which is synchronous, only holds when there are pending requests in all the connector ports. Dually, the merger merges data coming from ports a and b to port c, through j. Data merging is synchronous but only on two ends at each time: either on a and c or on b and c. This means that node j performs a non-deterministic choice whenever all all boundary ports have pending requests, thus preventing the firing of one of the input ports. The router connector, usually depicted as $\bigotimes$, is a mutual exclusive router of data, taking data from input port a into either b or c, depending on the existence of pending requests. When there are pending requests in both output ports, k chooses non-deterministically which of group of ports will synchronously fire: either a and b or a and c.

Some connectors exhibit *context-dependent* behaviour. The lossy channel is a typical example: whether it loses data or not depends exclusively on the environment issuing or not I/O requests at its boundary nodes. The router is another interesting case of a connector with both context-dependent and non-deterministic behaviour. Actually, context-dependency is a desired characteristic of some Reo channels, and as such, required to be correctly captured by any Reo semantic model.

Stochastic Reo [3, 36] extends Reo by adding non-negative real (stochastic) values to both channels and their ends to represent, respectively, *processing delays* and I/O *arrival rates*. The former models the time needed for the channel to internally process/conduct data from one point to another. A point in this context is either a channel end, a buffer or a null space where data is lost or automatically produced. One channel may be annotated with more than one processing delay, depending on their operational behaviour. On the other hand, I/O *arrival rates* model the time between consecutive arrivals to a channel end of environment-issued I/O requests. Figure 3 shows the basic channels of stochastic Reo, represented as normal Reo channels, but annotated with stochastic values.

Stochastic Reo is still compositional. Each composed channel retains its processing delay stochastic value. The request arrival rates, however, are only preserved at the boundary nodes of the connector. Once mixed nodes are internal (hidden from the exterior) the arrival request rates associated to the incident channel ends are ignored (*c.f.*, lossyfifo connector in Figure 3), meaning that these nodes are always ready to read/write data from/to the channels, a behaviour popularised by the metaphor of a *self-contained pumping station*, firstly mentioned in [3].

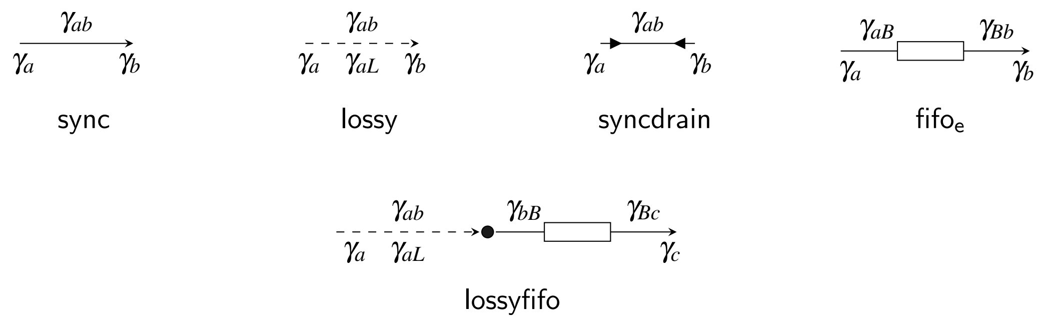


Figure 3: Primitive stochastic `Reo` channels.

2.2 Interactive Markov chains

Interactive Markov Chains (`IMC`) [24, 25] were initially proposed as a model for performance evaluation of distributed systems, combining continuous-time Markov chains (`CTMC`) [9, 6] and process algebra [7].

Therefore, an `IMC` evolves through tow kinds of transitions: *interactive* and *Markovian*. Interactive transitions capture the system's interaction with the environment through actions, as in process algebra. Their occurrence is assumed to be atomic (i.e., with no duration), since they are externally triggered. τ-transitions are a special case, representing internal (unobservable) activities. They are assumed to take place immediately, since no interaction with the environment is required. Markovian transitions, on the other hand, model a random delay in the system's evolution governed by a negative exponential distribution with a parameter $\gamma \in \mathbb{R}^+$. The introduction of this second type of transitions smoothly extends classical labelled transition systems, bringing into scene continuous time and specifying the delay probability for a state to change. Formally,

Definition 1. *An `IMC` is a tuple $I = (S, Act, \dashrightarrow, \longrightarrow, s)$, where S is a nonempty set of states; Act is a set of actions; $\dashrightarrow \subseteq S \times Act \times S$ is the interactive transition relation; $\longrightarrow \subseteq S \times \mathbb{R}^+ \times S$ is the Markovian transition relation; and $s \in S$ is the initial state.*

A Markovian transition $s \xrightarrow{\gamma} s'$ abstracts a move from state s to state s' with probability $1 - e^{-\gamma . t}$, where t is the amount of time units within which the transition must occur. Interactive transitions (s, a, s') are denoted as $s \overset{a}{\dashrightarrow} s'$ and represent a change in the system from state s to state s' through an external (single) action a that may be executed either immediately or blocked until the environment triggers it. Internal (interactive) transitions — τ-transitions — play an important role on an `IMC`. Since they do not interact with the environment, they are not delayed. Therefore, if an internal and a Markovian transition share the same source state, the internal transition will be fired before the Markovian, because the probability of the latter executing within 0 time units is always null: $1 - e^{-\gamma . 0} = 1 - e^0 = 1 - 1 = 0$. This fact is known as the *maximal progress assumption* and, in practice, has the effect of reducing the `IMC` size, by restricting both its state space and the number of transitions. Notice, however, that this only concerns Markovian transitions; interactive transitions may as well execute immediately. For further reference, states with outgoing τ-transitions are classified as *unstable* (*stable*, otherwise).

3 Interactive Markov chains for `Stochastic Reo`

This section introduces a semantic model for `Stochastic Reo` based on interactive Markov chains. In order to capture the behaviour of `Reo`, the underlying state space, as well as the labels used in interactive transitions, are suitably structured. Composition builds on the usual parallel composition for IMC, but discharging, through a synchronisation operator, all transitions which are not compliant with Reo operational semantics. This two-step composition is very much in the same spirit as the one defined for Reo automata [16].

Before introducing our proposal of an IMC model for stochastic Reo, referred to as $\mathtt{IMC}_{\mathtt{Reo}}$ in the sequel, some remarks on what a transition and a state represent in this context are in order to build up intuition.

As expected, states capture the possible behaviour of a connector, *i.e.*, data arrivals and data flowing through ports. A set of node/port names, $\mathscr{N}$, and a set of internal state names, $\mathscr{Q}$, are assumed. Thus the state of a $\mathtt{IMC}_{\mathtt{Reo}}$ model comprises three components — (R,T,Q) — where $R,T \in \mathscr{P}(\mathscr{N})$ denote, respectively, the set of requests and transmissions, *i.e.*, the sets of ports with, respectively, pending requests and data being transmitted; and $Q \in \mathscr{Q}$ is an internal state identifier. The latter is used to distinguish between control states in state-based connectors. For example, in the fifo it may indicate whether the buffer is empty or full, making $\mathscr{Q} = \{\mathsf{empty}, \mathsf{full}\}$. The selectors for a $\mathtt{IMC}_{\mathtt{Reo}}$ state s are denoted by R_s, T_s, Q_s, respectively.

Markovian transitions are labelled by $\gamma \in \mathbb{R}^+$. The negative exponential distribution parameter, γ, encodes, in each case, the connector processing delays and the rates of data arrival at its ports.

Interactive transitions, on their turn, are labelled with a set F of ports which, when firing, allow data to flow through them. Such ports correspond, therefore, to the set of actions observable at the relevant IMC state. The decision to take sets of actions (rather than a single action) to label interactive transitions is crucial to correctly capture synchrony in the semantics of Reo. In fact, ports firing synchronously to enable data flow are the rule rather than the exception in Reo.

In a nutshell, an $\mathtt{IMC}_{\mathtt{Reo}}$ modelling a Reo channel, is an instance of a classical IMC with a structured set of states and labels. Formally,

Definition 2. *An $\mathtt{IMC}_{\mathtt{Reo}}$ is a tuple (S, Act, $\dashrightarrow$, $\longrightarrow$, s), where $S \subseteq \mathscr{P}(\mathscr{N}) \times \mathscr{P}(\mathscr{N}) \times \mathscr{Q}$ is a nonempty set of states; $Act \subseteq \mathscr{P}(\mathscr{N})$ is a set of actions; $\dashrightarrow \subseteq S \times Act \times S$ is the interactive transition relation; $\longrightarrow \subseteq S \times \mathbb{R}^+ \times S$ is the Markovian transition relation; and $s \in S$ is the initial state.*

Markovian transitions (s,γ,s') are written as $s \xrightarrow{\gamma} s'$; whereas notation $s \overset{a_1 a_2 \cdots}{\dashrightarrow} s'$ is used for interactive transitions $(s,\{a_1,a_2,...\},s')$. An interactive transition with an empty set of actions is said to be unobservable and is denoted by $s \overset{\tau}{\dashrightarrow} s'$.

States of the form $(R,\emptyset,Q)$ are referred to as *request* states and depicted as $\boxed{R}_Q$; states of the form $(\emptyset,T,Q)$ are referred to as *transmission* states and depicted as $\{T\}_Q$; states of the form (R,T,Q) are called as *mixed* states and are rdepicted as $\boxed{R}\{T\}_Q$; finally, states of the form $(\emptyset,\emptyset,Q)$ are represented as $\emptyset_Q$ and denote the absence of both requests and data transmissions. For all representations, the buffer qualifier Q may be omitted, whenever clear from the context.

Figure 4 depicts the $\mathtt{IMC}_{\mathtt{Reo}}$ models corresponding to the basic stochastic Reo channels. To simplify the picture transition overlapping is avoided by the graphical replication of states suitably annotated with a dashed circle.

The $\mathtt{IMC}_{\mathtt{Reo}}$ model of a stochastic sync channel is interpreted as follows: initially, no requests are pending neither in port a nor in port b. Request arrive at port a (respectively, b) at rate γ_a (respectively, γ_b). The channel *blocks* until a request arrives to the other port. When state $\boxed{a,b}$ is reached, representing a configuration in which both ports have pending requests, then both may fire. That is, actions a and b may be activated simultaneously. At this moment, the channel starts transmitting data between a and b and evolves back to the initial state with a delay rate of γ_{ab}. For a syncdrain the interpretation is similar. The $\mathtt{IMC}_{\mathtt{Reo}}$ model of a stochastic lossy channel is also similar. However it exhibits two additional transitions to model the possibility of losing data: at state $\boxed{a}$, port a may fire, because there is no pending request at port b. This state captures the context-dependent behaviour characteristic of this channel. The $\mathsf{fifo}_{\mathsf{e}}$

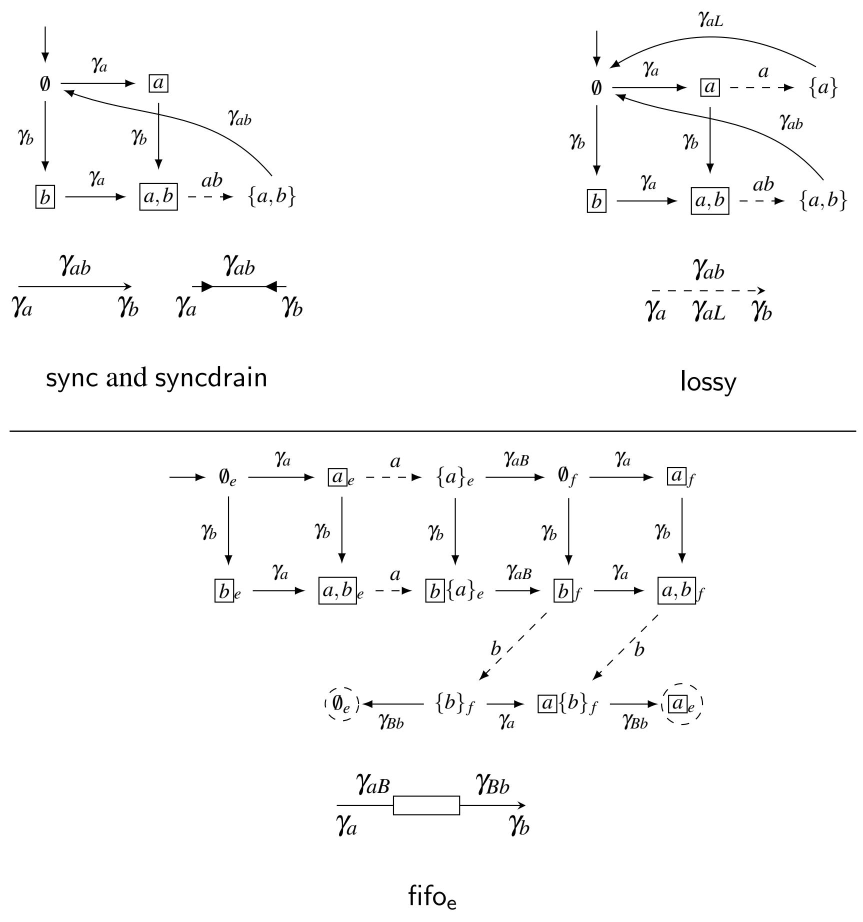


Figure 4: IMC for the basic stochastic Reo channels.

stochastic channel differs from the formers by introducing an internal state. Notice how pending requests at port a automatically fire when the *buffer* is empty (states $\boxed{a}_e$ and $\boxed{a,b}_e$), and requests at port b block until it is full (states $\boxed{a,b}_e$ and $\boxed{b}\{a\}_e$). Also, notice that, to maintain consistency, the internal state of this channel only changes after Markovian transitions representing processing delays succeed. Actually, this will be the rule in IMC$_{\text{Reo}}$ models.

4 New connectors from old

Reo connectors are composed through aggregation and sharing of border nodes. This section formalises this mechanism as IMC$_{\text{Reo}}$ combinators and proves their compositionality.

4.1 Parallel composition

Our starting point is an adaptation of the usual definition of IMC parallel composition, dealing explicitly with sets of actions.

Definition 3. *Let* $I = (S_I, Act_I, \dashrightarrow_I, \longrightarrow_I, s_i)$ *and* $J = (S_J, Act_J, \dashrightarrow_J, \longrightarrow_J, s_j)$ *be two* IMC$_{\text{Reo}}$ *models.*

The parallel composition of I and J with respect to a set of actions M is defined as

$$I \parallel_M J = (S, Act, \dashrightarrow, \longrightarrow, (s_i, s_j))$$

where $S = S_I \times S_J$, $Act = Act_I \cup Act_J$, and $\dashrightarrow$ and $\longrightarrow$ are the smallest relations satisfying

1. *If $i_1 \overset{A_I}{\dashrightarrow}_I i_2$ and $A_I \cap M = \emptyset$, then $(i_1, j) \overset{A_I}{\dashrightarrow} (i_2, j)$, for $j \in S_J$.*
2. *If $j_1 \overset{A_J}{\dashrightarrow}_J j_2$ and $A_J \cap M = \emptyset$, then $(i, j_1) \overset{A_J}{\dashrightarrow} (i, j_2)$, for $i \in S_I$.*
3. *If $i_1 \overset{A_I}{\dashrightarrow}_I i_2$, $j_1 \overset{A_J}{\dashrightarrow}_J j_2$ and $(A_I \cap A_J) \subseteq M$, and $A_I, A_J \neq \emptyset$ then $(i_1, j_1) \overset{A_I \cup A_J}{\dashrightarrow} (i_2, j_2)$.*
4. *If $i_1 \xrightarrow{\gamma}_I i_2$, then $(i_1, j) \xrightarrow{\gamma} (i_2, j)$, for $j \in S_J$,*
5. *If $j_1 \xrightarrow{\gamma}_J j_2$, then $(i, j_1) \xrightarrow{\gamma} (i, j_2)$, for $i \in S_I$,*

The first three clauses in the definition above deal with interactive transitions: the first two tackle the independent evolution of each connector; the third one addresses their (synchronous) joint evolution. Clauses 4. and 5. deal with Markovian transitions which are always interleaved. Figure 5 depicts a fragment of the IMC resulting from the parallel composition of a lossy and a fifo$_\text{e}$ channel with respect to set $M = \{b\}$ of shared nodes. A pair of states (i, j) is depicted as $i|j$. The complete IMC has 72 states and 182 transitions.

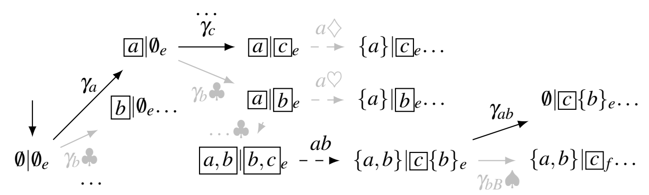

γ_c
γ_a
γ_b♣
γ_{ab}
γ_{bB}♠
a♢
a♡
ab
$\emptyset|\emptyset_e$
$\boxed{a}|\emptyset_e$
$\boxed{b}|\emptyset_e\ldots$
$\boxed{a}|\boxed{c}_e$
$\boxed{a}|\boxed{b}_e$
$\boxed{a,b}|\boxed{b,c}_e$
$\{a\}|\boxed{c}_e\ldots$
$\{a\}|\boxed{b}_e\ldots$
$\{a,b\}|\boxed{c}\{b\}_e$
$\emptyset|\boxed{c}\{b\}_e\ldots$
$\{a,b\}|\boxed{c}_f\ldots$

Figure 5: Parallel composition of a lossy and fifo$_\text{e}$.

Note that clause 3 of Definition 3 does not capture the joint evolution of τ-transitions (which is precluded by condition $A_I, A_J \neq \emptyset$). Actually, this is just a design decision in accordance to the homologous definition in [24]. We could otherwise allow remove the condition and allow for the joint evolution of two or more τ-transitions. In practice, this would result in an extra internal transition; however the resulting IMC$_{\text{Reo}}$ chains would be equivalent (at least in a weaker way) due to the maximal progress assumption.

To illustrate both alternatives, consider the composition of two double − fifo$_\text{e}$ connectors. A fragment of the IMC$_{\text{Reo}}$ model for each double − fifo$_\text{e}$ is depicted in Figure 12. Let us focus on the joint state $\emptyset_f|\emptyset_e$, assuming, for simplicity, their piecewise union. Figure 6 shows the configuration of the two channels, at that state, alongside with a fragment of the corresponding IMC$_{\text{Reo}}$ models, which we want to compose in parallel for the set of ports $\{a, c\}$.

Applying the parallel composition, as given by Definition 3, will create the IMC$_{\text{Reo}}$, a fragment of which is depicted in solid black in Figure 7. State $\emptyset_{fe}|\emptyset_{fe}$ has four Markovian transitions (from the application of clauses 4 and 5) and two τ-transitions (from clauses 1 and 2). If, otherwise, the alternative version of clause 3 is assumed, without the $A_I, A_J \neq \emptyset$ condition, the extra transition $\emptyset_{fe}|\emptyset_{fe} \overset{\tau}{\dashrightarrow} \{b\}_{fe}|\{d\}_{fe}$, depicted in grey in Figure 7 would be possible.

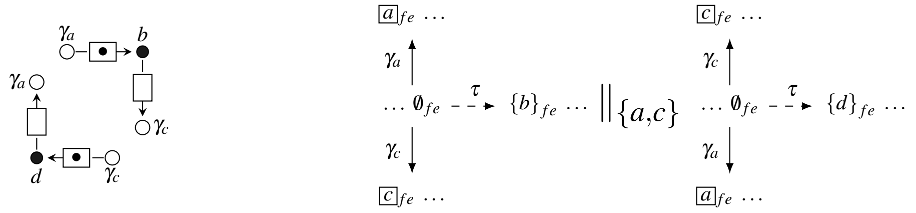


Figure 6: Composing two $\mathsf{double-fifo_e}$ connectors.

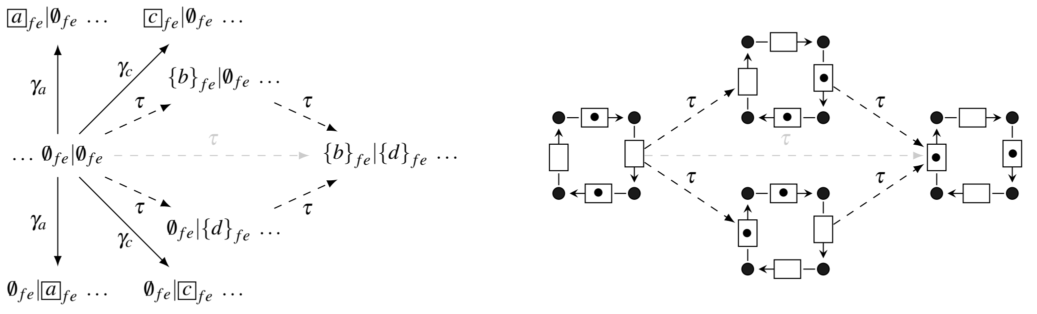

γ_a
γ_c
τ
$\emptyset_{fe}|\emptyset_{fe}$
$\{b\}_{fe}|\emptyset_{fe}$
$\{b\}_{fe}|\{d\}_{fe}$
$\emptyset_{fe}|\{d\}_{fe}$

Figure 7: Parallel composition of two $\mathsf{double-fifo_e}$ (fragment) and the corresponding circuit evolution.

4.2 Synchronisation

The definition of parallel composition, however, has to be adjusted to correctly capture the intended semantics for channel composition in `Reo`. The mismatch concerns `Reo` mixed nodes which are not supposed to actively block behaviour, rather acting like *self-contained pumping stations*, in the popular metaphor of [3]. Failing to take this into account generates unwanted behaviour, making the semantics unsound. Consider, for example, the composition of a lossy and a $\mathsf{fifo_e}$ partially depicted in Figure 5. The presence of transition $\boxed{a}|\boxed{b}_e \overset{a}{\dashrightarrow} \{a\}|\boxed{b}_e$ shows that in the composed connector a data token arriving to port a could be lost, even when the buffer is empty. This possibility, however, violates the mixed node assumption in the `Reo` *rationale* in the sense that transitions cannot occur from a state which *actively blocks* a mixed node. The following definition captures this notion of *active blocking*.

Definition 4. *Given an* IMC$_{\mathtt{Reo}}$ *model over a composed state space* $(S_1 \times S_2, Act, \dashrightarrow, \longrightarrow, s)$, *a state* (i,j) actively blocks *a set of nodes* M *if there exists a transition* $(i,j) \overset{X}{\dashrightarrow} (\_,\_)$ *with* $X \cap M = \emptyset$, $R_i \cap M = \emptyset$ *and* $R_j \cap M \neq \emptyset$, *or vice-versa.*

Clearly, in the example above, state $\boxed{a}|\boxed{b}_e$ actively blocks the (singleton) set of mixed nodes $M = \{b\}$ in the composition of the two connectors.

We are now ready to introduce a synchronisation operation to prune this sort of unwanted transitions. Notation $i{\restriction_M}$, given a state $i = (R,T,Q)$ and a set of ports M, identifies a restricted state from i in which all ports in M are hidden (*i.e.*, $i{\restriction_M} = (R \setminus M, T, Q)$), and therefore removed from the set of active requests in i. This extends to composite states (i,j) in the obvious way: $(i,j){\restriction_M} = (i{\restriction_M}, j{\restriction_M})$.

Definition 5. *Let* $I = (S_1 \times S_2, Act, \dashrightarrow, \longrightarrow, s)$ *be an* IMC$_{\mathtt{Reo}}$ *model over a composite state space, and* M *a set of nodes. The synchronisation of* I *with respect to* M *is given by*

$$\partial_M I = (S_M, Act \setminus M, \dashrightarrow_M, \longrightarrow_M, s)$$

where $S_M = \{(i,j)\upharpoonright_M \mid (i,j) \in S_1 \times S_2\}$ and $\dashrightarrow_M$ and $\longrightarrow_M$ are the smallest relations satisfying, respectively, conditions 1 and 2 below:

1. *If $(i,j) \overset{X}{\dashrightarrow} (i',j')$, and (i,j) does not actively block M, then $(i,j)\upharpoonright_M \overset{X\backslash M}{\dashrightarrow}_M (i',j')\upharpoonright_M$.*

2. *If $(i,j) \xrightarrow{\gamma} (i',j')$ and $(R_{i'} \cup R_{j'}) \cap M = \emptyset$, then $(i,j)\upharpoonright_M \xrightarrow{\gamma}_M (i',j')\upharpoonright_M$.*

In Figure 5, interactive transitions that need to be deleted because their origin states are actively blocking a set of mixed nodes, are labeled with $\heartsuit$. One such transition is $\boxed{a}|\boxed{b}_e \overset{a}{\dashrightarrow} \{a\}|\boxed{b}_e$, where, as explained above, for $M = \{b\}$, $X = \{a\}$, clearly $X \cap M = \emptyset$, $R_{\boxed{a}} \cap M = \{a\} \cap M = \emptyset$, and $R_{\boxed{b}_e} \cap M$ holds $\{b\} \cap M \neq \emptyset$. That is, port b, which was expected to automatically fire, is blocked and only port a fires. On the other hand, Markovian transitions going to states with requests in the mixed nodes are labeled with $\clubsuit$. An example is transition $\emptyset|\emptyset_e \xrightarrow{\gamma_b} \boxed{b}|\emptyset_e$, where for $M = \{b\}$, $(R_{\boxed{b}} \cup R_{\emptyset_e}) \cap M \neq \emptyset$.

Composition in $\mathtt{IMC}_{\mathtt{Reo}}$ can finally be defined resorting to both parallel and synchronisation,

Definition 6. *The composition of $\mathtt{IMC}_{\mathtt{Reo}}$ models I and J with respect to a set of actions M is given by*

$$\partial_M(I_1 \;\|_M\; I_2)$$

4.3 Properties

This section discusses a number of properties of these operators with respect to (strong) bisimilarity. We start by defining the latter as a slight extension of standard IMC bisimilarity [24] to sets of labels.

Definition 7 (Strong bisimulation)**.** *Let $I = (S, Act, \dashrightarrow, \longrightarrow, s)$ be an $\mathtt{IMC}_{\mathtt{Reo}}$ model. An equivalence relation $\mathscr{R} \subseteq S \times S$ is a strong bisimulation if for any $(i,j) \in \mathscr{R}$ and equivalence class $C \in S/\mathscr{R}$ the following holds:*

1. *for each $A \subseteq Act$ and some $i' \in C$, if $i \overset{A}{\dashrightarrow} i'$ then there exists a state $j' \in C$ such that $j \overset{A}{\dashrightarrow} j'$, and $(i',j') \in \mathscr{R}$, and vice-versa;*

2. *if i is a* stable *state then $\Gamma(i,C) = \Gamma(j,C)$;*

where $\Gamma : S \times \mathscr{P}(S) \to \mathbb{R}^+$ is a function that cumulates rates specified to reach a set of states Q from a state i: $\Gamma(i,Q) = \Sigma_{i' \in Q}[\gamma \mid i \xrightarrow{\gamma} i']$.

Two $\mathtt{IMC}_{\mathtt{Reo}}$ models I and J with disjoint state spaces S_I and S_J and initial states i and j, respectively, are *strong bisimilar* (denoted $I \sim J$) if there exists a strong bisimulation $\mathscr{R}$ on $S_I \cup S_J$ such that $(i,j) \in \mathscr{R}$. Strong bisimilarity is the largest strong bisimulation, which is as expected an equivalence relation.

Theorem 1. *Let I, I_1, I_2 and I_3 be $\mathtt{IMC}_{\mathtt{Reo}}$ models, where Act_1 is the alphabet of I_1 and M,N are sets of mixed nodes. The following holds:*

1. $\partial_M(I_1 \;\|_N I_2) \sim I_1 \;\|_N \partial_M I_2,\;$ *if* $Act_1 \cap M = \emptyset$

2. $\partial_N(\partial_M I) = \partial_M(\partial_N I) = \partial_{M \cup N} I$

3. $(I_1 \|_M I_2) \|_M I_3 \sim I_1 \|_M (I_2 \|_M I_3)$

4. $I_1 \|_M I_2 \sim I_2 \|_M I_1$

Proof. For 1. note that transitions in $\partial_M(I_1 \;\|_N I_2)$ can be of one of the following forms:

- a M restriction of a transition $(i,j) \overset{X}{\dashrightarrow} (i',j)$, such that $i \overset{X}{\dashrightarrow} i'$ exists in I_1, $X \cap N = \emptyset$ and (i,j) does not actively block M.
- a M restriction of a transition $(i,j) \overset{X}{\dashrightarrow} (i,j)$, such that $j \overset{X}{\dashrightarrow} j'$ exists in I_2, $X \cap N = \emptyset$ and (i,j) does not actively block M.
- a M restriction of a transition $(i,j) \overset{X_1 \cup X_2}{\dashrightarrow} (i',j')$, such that $i \overset{X_1}{\dashrightarrow} i'$ and $j \overset{X_2}{\dashrightarrow} j'$ exists in I_1 and I_2, respectively, $X_1 \cap X_2 \subseteq N$ and (i,j) does not actively block M.

All such transitions are possible for $I_1 \;\|_N\; \partial_M I_2$, but it remains to show that in neither of them can the origin state block M. Consider the first type of transitions (the argument for the others are similar). Note that there are only two ways of state (i,j) being able to block M. The first possibility is $X \cap M = \emptyset$, which is the case: $R_i \cap M = \emptyset$ because $Act_1 \cap M = \emptyset$, and $R_j \cap M \neq \emptyset$, which is false because all labels in any transitions in $\partial_M I_2$ do not contain elements of M. The second alternative requires $X \cap M = \emptyset$ again, and $R_j \cap M = \emptyset$, which holds as discussed above, but also $R_i \cap M \neq \emptyset$ which is false by hypothesis. Therefore, allowed transitions in $I_1 \;\|_N\; \partial_M I_2$ do not block M. In principle, it could exhibit more interactive transitions than $\partial_M(I_1 \;\|_N I_2)$, namely transitions in I_1 with elements in M. This cannot be the case, however, because of the assumption $Act_1 \cap M = \emptyset$.

In turn, Markovian transitions in $\partial_M(I_1 \;\|_N I_2)$ can be of one of the following forms:

- a M restriction of a transition $(i,j) \xrightarrow{\gamma} (i',j)$, such that $i \xrightarrow{\gamma} i'$ exists in I_1 and $(R_{i'} \cup R_j) \cap M = \emptyset$.
- a M restriction of a transition $(i,j) \xrightarrow{\gamma} (i,j')$, such that $j \xrightarrow{\gamma} j'$ exists in I_2 and $(R_i \cup R_{j'}) \cap M = \emptyset$.

Both cases also occur in $I_1 \;\|_N\; \partial_M I_2$; it remains to show that the request sets of the target states of these transitions do not have elements in M. In fact, assumption $Act_1 \cap M = \emptyset$ implies that Markovian transitions in I_1 cannot have target states i' with $R_{i'} \cap M \neq \emptyset$; and, by definition, for all the target states j' of Markovian transitions in $\partial_M I_2$, $R_{j'} \cap M = \emptyset$ holds. Hence, Markovian transitions in $I_1 \;\|_N\; \partial_M I_2$ are the same as in $\partial_M(I_1 \;\|_N I_2)$, which preserves the cumulation of rates.

For 2. it is enough to note that in both sides of the bisimilarity equation interactive and Markovian transitions are restricted to $M \cup N$, and also that interactive transitions in I whose origin state does not actively block neither M nor N do the same for $M \cup N$.

Associativity and commutativity of $\|_M$ are easy to prove: for Markovian transitions strict interleaving applies and therefore the contribution of each chain is always preserved; for interactive transitions, when in presence of sharing, associativity and commutativity of $\cup$ entail 3. and 4., respectively. □

Theorem 2 (Substitutability for $\|_M$)**.** *Let I_1,I_2 and I_3 be IMC$_{Reo}$ models, and M be a set of nodes, such that $I_1 \sim I_3$. Then, $I_1 || _M I_2 \sim I_3 || _M I_2$.*

Proof. Every interactive transition of I_1 is exactly matched by an equally labelled transition in I_3, because $I_1 \sim I_3$. Therefore, the contribution of both I_1 and I_3 for interactive transitions of their parallel composition with I_2 is the same. For Markovian transitions $I_1 \sim I_3$ guarantees that for bisimilar states the cumulative rate in both chains is the same. In the parallel composition Markovian transitions of I_1 are interleaved with those of I_2. Once a state of I_1 is replaced by a bisimilar one, the cumulative rates are not affected, even if the number of Markovian transitions may differ. □

This result establishes substitutability for parallel composition with respect to a set of nodes M. A similar result for ∂_M holds only for a stricter notion of equivalence. Actually, synchronisation prunes a number of transitions which depend on data requests active in each state. This entails the need to incorporate this sort of information in a suitable definition of behavioural equivalence.

Definition 8 (Behavioural equivalence)**.** *Two IMC$_{Reo}$ models I and J are* behaviourally equivalent*, denoted by $I \equiv J$, if they are strong bisimilar and for each pair (i,j) of related states, their sets of requests are equal: $R_i = R_j$.*

Clearly $\equiv \subseteq \sim$. With this coarser equivalence substitutability for ∂_M can be established.

Theorem 3 (Substitutability for ∂_M)**.** *Let I_1 and I_2 be two $\texttt{IMC}_{\texttt{Reo}}$ models , and M a set of nodes, such that $I_1 \equiv I_2$. Then, $\partial_M I_1 \equiv \partial_M I_2$.*

Proof. An interactive transition in $\partial_M I_1$ is the M restriction of a I_1 interactive transition, say $(i,j) \overset{X}{\dashrightarrow} (i',j')$ such that (i,j) does not actively block M. As $I_1 \sim I_2$ there exists a transition $(k,l) \overset{X}{\dashrightarrow} (k',l')$ in I_2 for bisimilar states (i,j) and (k,l). The extra condition in the definition of $\equiv$ further imposes that $R_{(i,j)} = R_{(k,l)}$, which, both being composed states, has to be stated piecewise, i.e., $R_i = R_k$ and $R_j = R_l$. But this means that if the original transition in $\partial_M I_1$ does not block M, this one in $\partial_M I_2$ has the same property. A similar argument applies to the analysis of Markovian transitions. Actually, for stable, bisimilar states (i,j) and (k,l) in I_1 and I_2, respectively, and each bisimilarity equivalence class C, $\Gamma((i,j),C) = \Gamma((k,l),C)$. This equality does not necessarily holds when comparing $\partial_M I_1$ and $\partial_M I_2$ because the definition of synchronisation considers only (M restrictions of) Markovian transitions $(i,j) \xrightarrow{\gamma} (i',j')$ such that $(R_{i'} \cup R_{j'}) \cap M = \emptyset$ and so, unless $R_{i'} = R_{k'}$ and $R_{j'} = R_{l'}$, one can not conclude that the corresponding cumulative rates $\Gamma((i',j'),C), \Gamma((k',l'),C)$ are equal, because different markovian transitions could have been pruned in $\partial_M I_1$ and $\partial_M I_2$. But such is the case, however, if $I_1 \equiv I_2$. $\square$

4.4 Cleaning up unintended transitions

In general, when `Reo` channels are set in parallel within a connector, they evolve independently. However, when connected through their ends, data flows from one to the other, in sequence, and there is a clear intended flow *direction*. Until now, however, we have refrained from explicitly modelling the difference between input and output ports, which are responsible for setting the data flow direction. This may generate unintended transitions. In Figure 5, for example, node $\{a,b\}|\boxed{c}\{b\}_e$ evolves interleaved via γ_{ab} or γ_{bB} to the same state. However, this leads to an artificial configuration in which the buffer becomes full before data is transmitted through the lossy channel. Moreover, when a channel is transmitting data from one port to another, arrival of requests might be undesirable, as ports are busy.

Definition 9. *Let $M \subseteq \mathscr{N}$ and $I = (S, Act, \dashrightarrow, \longrightarrow, s)$ be an $\texttt{IMC}_{\texttt{Reo}}$ model. Suppose there exists a relation $<$ on $\mathscr{N}$ such that $a < b$ when data flows from a to b, with $a,b \in \mathscr{N}$. The cleaning of I with respect to M, denoted $\mathscr{C}_M I$, corresponds to restricting $\partial_M I$ so that for all its Markovian transitions $i \xrightarrow{\gamma} f$*

1. *$R_f \cap AN(i) = \emptyset \wedge R_i \subset R_f$, where $AN(i) = T_i \cup \{j \mid \exists_{k \in T_i} \cdot j < k \wedge k < j\}$;*

2. *$T_f \subset T_i$;*

hold, and all its interactive transitions $j \overset{X}{\dashrightarrow} k$ respect the following condition

3. *$\neg\exists_{j \overset{Y}{\dashrightarrow} l \in \dashrightarrow} \cdot X = Y \wedge T_k \cap M = \emptyset \wedge T_l \cap M \neq \emptyset$.*

Informally, an $\texttt{IMC}_{\texttt{Reo}}$ model failing to respect condition 1. allows requests on nodes that are actively transmitting data. On the other hand, whenever condition 2. does not hold, the model fails to preserve transmission sequencing. Finally, if condition 3. fails, nondeterminism is enforced where it should not exist.

Illustration of conditions 2. and 3. can be found in Figure 5. Note, for example, that the transition marked with ♠ does not respect transmission sequencing: $\{a,b\}|\boxed{c}\{b\}_e \xrightarrow{\gamma_{bB}} \{a,b\}|\boxed{c}_f$. Actually $T_i = T_{\{a,b\}|\boxed{c}\{b\}_e} = \{a,b\}$ and $T_f = T_{\{a,b\}|\boxed{c}_f} = \{a,b\}$, so $T_i \not\subset T_f$.

On the other hand, the transition marked with $\diamondsuit$ enforces nondeterminism. Note that state $\boxed{a}\|\boxed{c}$ is equal to $\boxed{a,b}\|\boxed{b,c}$, modulo $\restriction_M$, for $M = \{b\}$. Therefore, after synchronisation the following transitions coexist: $\boxed{a}|\boxed{c} \overset{a}{\dashrightarrow} \{a\}|\boxed{c}$ and $\boxed{a}|\boxed{c} \overset{a}{\dashrightarrow} \{a,b\}|\boxed{c}\{b\}$. Nondeterminism is enforced because both transitions are equally labelled and $T_{\{a\}|\boxed{c}} \cap M = \emptyset$ and $T_{\{a,b\}|\boxed{c}\{b\}} \cap M \neq \emptyset$.

The composition illustrated in Figure 5 is revisited in Figure 8 after synchronisation and cleaning.

$$\begin{tikzcd}
{} \arrow[r] & \emptyset|\emptyset_e \arrow[r, "\gamma_a"] \arrow[d, phantom, "\vdots"] & \boxed{a}|\emptyset_e \arrow[r, "\gamma_c"] \arrow[d, phantom, "\vdots"] & \boxed{a}|\boxed{c}_e \arrow[r, dashed, "a"] & \{a,b\}|\boxed{c}\{b\}_e \arrow[r, "\gamma_{ab}"] & \emptyset|\boxed{c}\{b\}_e \ldots \\
& {} & {} & & &
\end{tikzcd}$$

Figure 8: Parallel composition of a lossy and fifo$_{\mathsf{e}}$ after synchronisation and cleaning.

A substitutability result for $\mathscr{C}_M$ can only be formulated in terms of a slightly coarser version of strong bisimilarity as follows.

Definition 10 (Time-independent strong bisimulation)**.** *Let $I = (S, Act, \dashrightarrow, \longrightarrow, s)$ be an $\texttt{IMC}_{\texttt{Reo}}$ model. An equivalence relation $\mathscr{R} \subseteq S \times S$ is a time-independent strong bisimulation if for any $(i, j) \in \mathscr{R}$ and equivalence class $C \in S/\mathscr{R}$ the following holds:*

1. *for each $A \subseteq Act$ and some $i' \in C$, if $i \xrightarrow{*} \overset{A}{\dashrightarrow} \xrightarrow{*} i'$, then there exists a state $j' \in C$ such that $j \xrightarrow{*} \overset{A}{\dashrightarrow} \xrightarrow{*} j'$, and $(i', j') \in \mathscr{R}$, and vice-versa,*

where $\xrightarrow{*}$ represents a possibly empty sequence of Markovian transitions.

Two $\texttt{IMC}_{\texttt{Reo}}$ models I and J with disjoint state spaces S_I and S_J and initial states i and j, respectively, are *time-independent strong bisimilar* (denoted $I \sim_{ti} J$) if there exists a time-independent strong bisimulation $\mathscr{R}$ on $S_I \cup S_J$ such that $(i, j) \in \mathscr{R}$.

Theorem 4. *Let I and J be two $\texttt{IMC}_{\texttt{Reo}}$ models, such that $I \sim J$. Then $I \sim_{ti} J$.*

Proof. From the structure of a $\texttt{IMC}_{\texttt{Reo}}$ model, it is obvious that each interactive transition may be preceded/followed by a (possibly empty) number of Markovian transitions. Thus, each interactive transition of I is of the form $i \xrightarrow{*} \overset{A}{\dashrightarrow} \xrightarrow{*} i'$, where i (respectively, i') is the target (respectively, the source) state of some interactive transition. The same goes to J: each interactive transition is of the form $j \xrightarrow{*} \overset{A}{\dashrightarrow} \xrightarrow{*} j'$, with j (respectively, j') being the target (respectively, the source) state of some interactive transition in J. Since $I \sim J$ (by assumption) then, i and j are bisimilar and so are i' and j'. This is exactly the case when $I \sim_{ti} J$. □

Theorem 5. *Let $M \subseteq \mathscr{N}$ and I and J be two $\texttt{IMC}_{\texttt{Reo}}$ models, such that $I \sim J$. Then $\mathscr{C}_M I \sim_{ti} \mathscr{C}_M J$*

Proof. From Theorem 4, $I \sim_{ti} J$. Therefore, removing Markovian transitions from I and J in the context of the cleaning operation does not affect the $\sim_{ti}$ relation. It is then only necessary to show that removing interactive transitions (that enforce nondeterminism) does not affect the relation as well.

In case that no interactive transition enforces nondeterminism in I (consequently there is also no such transitions in J, because $I \sim J$), then $I \sim_{ti} J$ holds trivially. In case that there is a transition $i \overset{A}{\dashrightarrow} i'$ enforcing nondeterminism in I, then there is also such a transition $j \overset{A}{\dashrightarrow} j'$ in J. Thus, they are both removed, in I and J, on cleaning. Possibly there remain Markovian transitions that have also to be removed as their source states become unreachable after the interactive transitions above are eliminated; but this will not affect the relation (because $I \sim_{ti} J$). Possible interactive transitions removed in I for the same reason are also removed in J. Thus, $\mathscr{C}_M I \sim_{ti} \mathscr{C}_M J$. □

Note, however, that $\mathscr{C}_M I \sim \mathscr{C}_M J$ when $I \sim J$, or even when $I \equiv J$, does not hold in general. Suppose, for example, that $I \sim J$, where after synchronisation I and J are as depicted in Figure 9.

After cleaning, transitions $\boxed{a}|\boxed{c} \overset{a}{\dashrightarrow} \{a\}|\boxed{c} \xrightarrow{0.5} \emptyset$ in I will disappear. Clearly, states $\{a,b\}|\boxed{c}\{b\}$ (in both chains) will not remain bisimilar since their commutative rates become 0.4 in I and 0.9 in J.

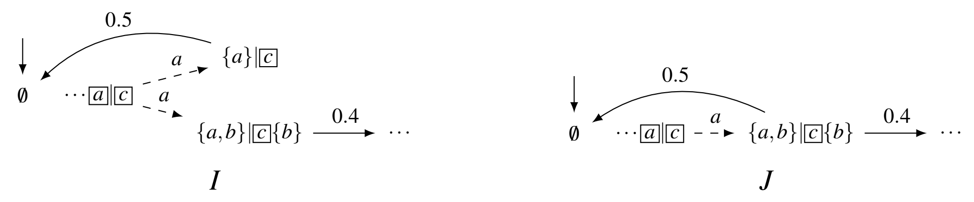


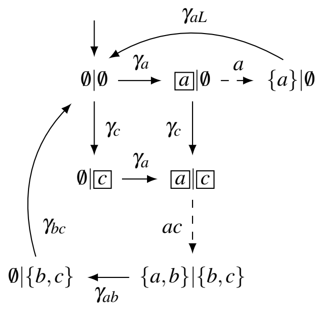
Figure 9: A counterexample.

4.5 Composition examples

To illustrate composition in IMC$_{\texttt{Reo}}$, this section discusses three different examples, each one with a specific particularity worth to point out.

Let us start with the composition of a lossy and a sync channel depicted in Figure 10. The result is an IMC$_{\texttt{Reo}}$ corresponding to a lossy channel with ports a and c and a processing delay rate $\phi(\gamma_{ab}, \gamma_{bc})$, which combines (via convolution) the exponential distributions modelled by γ_{ab} and γ_{bc} as an approximated exponential distribution.

Up to this rate the sync channel behaves as the identity for IMC$_{\texttt{Reo}}$ composition, as it happens in Reo. This is confirmed when comparing lossy and lossysync via time-independent strong bisimilarity.



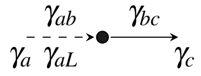


Figure 10: Composing a lossy and a sync channel.

Figure 11 depicts the composition of a lossy and a fifo$_{\mathsf{e}}$ channel, *i.e.*the lossyfifo connector first shown in Figure 3. Observe that data is not lost when the buffer is empty, as it may be the case in other semantic models for Reo unable to capture context dependency (see discussion in [16]). Rather data is lost only when the buffer is full and there is a request at the input port.

Finally, figure 12 shows a fragment of the composition of two fifo$_{\mathsf{e}}$ channels (the complete model has 39 states and 75 transitions). Note that when the first buffer is full and the second one empty, data may flow immediately to the latter, freeing the former. The τ-transitions, which appear after synchronisation, explicitly model this behaviour. Consequently, the *maximal progression assumption* can simplify the overall chain by deleting Markovian transitions leaving from such unstable states. The model would be reduced to approximately 35 states, depending on the bisimulation algorithm used for minimisation.

Note that the decision of modelling a two or more memory position fifo channel by the composition of several one-position fifo channels, although typical in the Reo *rationale* (obtain complex connectors from the composition of simpler ones), brings inefficiency to composition as the state space grows with the growth of memory positions. An alternative solution would be to regard fifo channels as normal queues with a bounded capacity. This would require, for instance, that states in IMC$_{\texttt{Reo}}$ were modelled as triples (R, T, n), where n stands for the number of free memory positions.

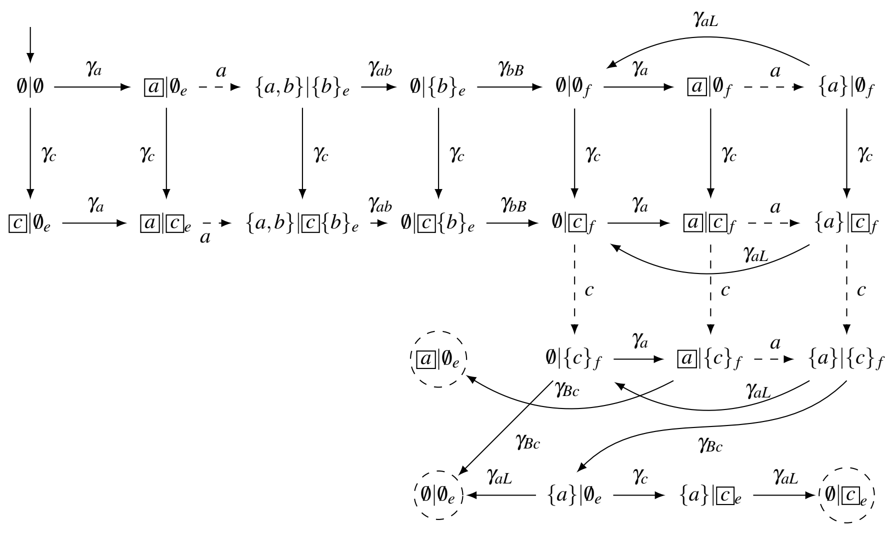


Figure 11: Composing a lossy and a $\mathsf{fifo_e}$ channel.

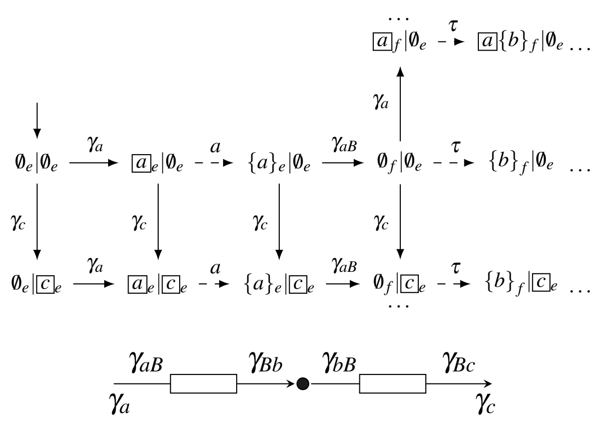


Figure 12: Composition of two $\mathsf{fifo_e}$ channels (fragment).

It can be argued that the examples above are simple in the sense that each node connects, at most, two channel ends, while in practice this number is often bigger. We assume, however, that composition is always performed in this binary way; when three or more channel ends have to be joined, the connector design is refactored to comply to this assumption through the use of mergers, replicators and routers which are given as stochastic primitive three-port channels.

Figure 13 (first row) presents the stochastic version (abstracted) of the replicator, the merger and the router connectors. The corresponding $\mathtt{IMC_{Reo}}$ models are given in Figure 14. These abstractions assume that the processing delays shown correspond to the composition of all delays involved in each relevant data flow path: in a replicator there is only one such path which involves all the nodes of the connector. In a merger there are two: one involving nodes a, j and c[1], and another involving nodes b, j and c. Finally, in a router there are also two paths: one involving nodes a, j, l, k and b and the other a, j, m, k and c.

The second row of Figure 13 illustrates how a connector can be transformed to comply to the as-

[1]These node names refer to the connectors depicted in Figure 2.

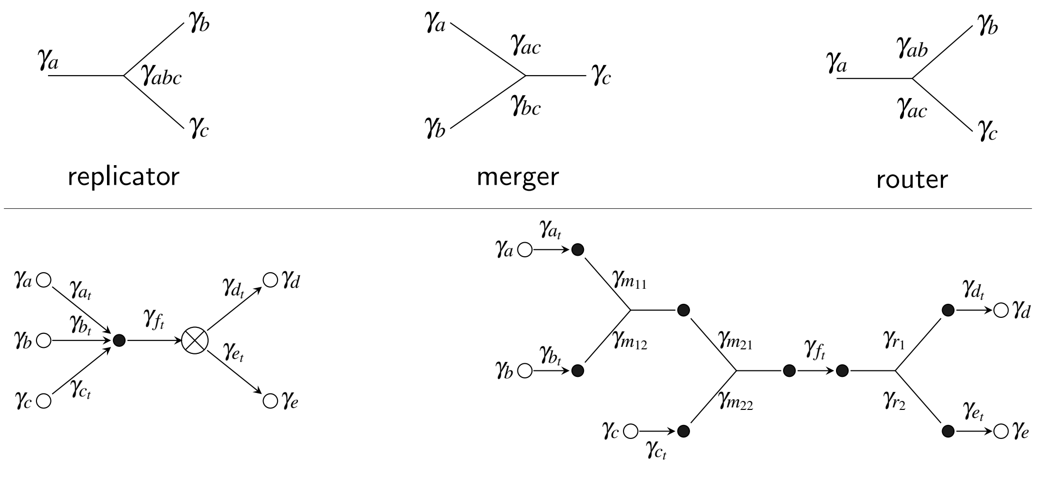


Figure 13: Stochastic three-port basic connectors and connector refactoring.

sumption that each node connects, at most, two channel ends. The processing delays in each three-port channel may be negligibly low to cope with the *self-contained pumping station* assumption of `Reo` nodes; otherwise, they shall model the delay that a node take to copy data or to make decisions.

5 A case study

This section discusses a (fragment of) a case study which is part of a broader analysis of the ASK system [34] with respect to performance and resource usage. A similar analysis conducted in Stochastic `Reo` is reported in [34] which makes possible a comparison with the approach proposed in this paper.

5.1 Tool support

The main motivation for using `IMC` as a semantic model for `Stochastic Reo` is the availability of well-tested tools, developed within the `IMC` community. These can be reused in a tool chain organised in two stages: one for developing $\mathtt{IMC}_{\mathtt{Reo}}$ models, another for their quantitative/qualitative analysis. Building $\mathtt{IMC}_{\mathtt{Reo}}$ models from `Stochastic Reo` connectors involves tree main stages: parallel composition, synchronisation and cleaning. With just paper and pencil this is a cumbersome process. Therefore, an extra tool to design connectors and systematically convert them into $\mathtt{IMC}_{\mathtt{Reo}}$ models was developed and is part of the tool chain mentioned above.

5.1.1 Modelling

The first element in the tool chain allows precisely for the design of complex connectors in `Reo`-like coordination models. It was proposed in [38, 37], and made available as an Eclipse plugin editor and a front-end for the `CooPLa` language [38]. The latter is the design language for connectors, introduced as reusable coordination patterns in `Reo` like formalisms. The editor is fully-functional, providing syntax highlighting, syntactic and semantic error reporting and intelligent code completion at edition-time. In addition, it incorporates a connector visualiser (as a graph of channels) that is assembled again at edition-time. A screenshot is shown in Figure 15. The editor is prepared for several source-to-source transformations which makes possible the translation of `CooPLa` patterns into `Reo` networks (preparing input for ECT [4]) or Vereofy models [8].

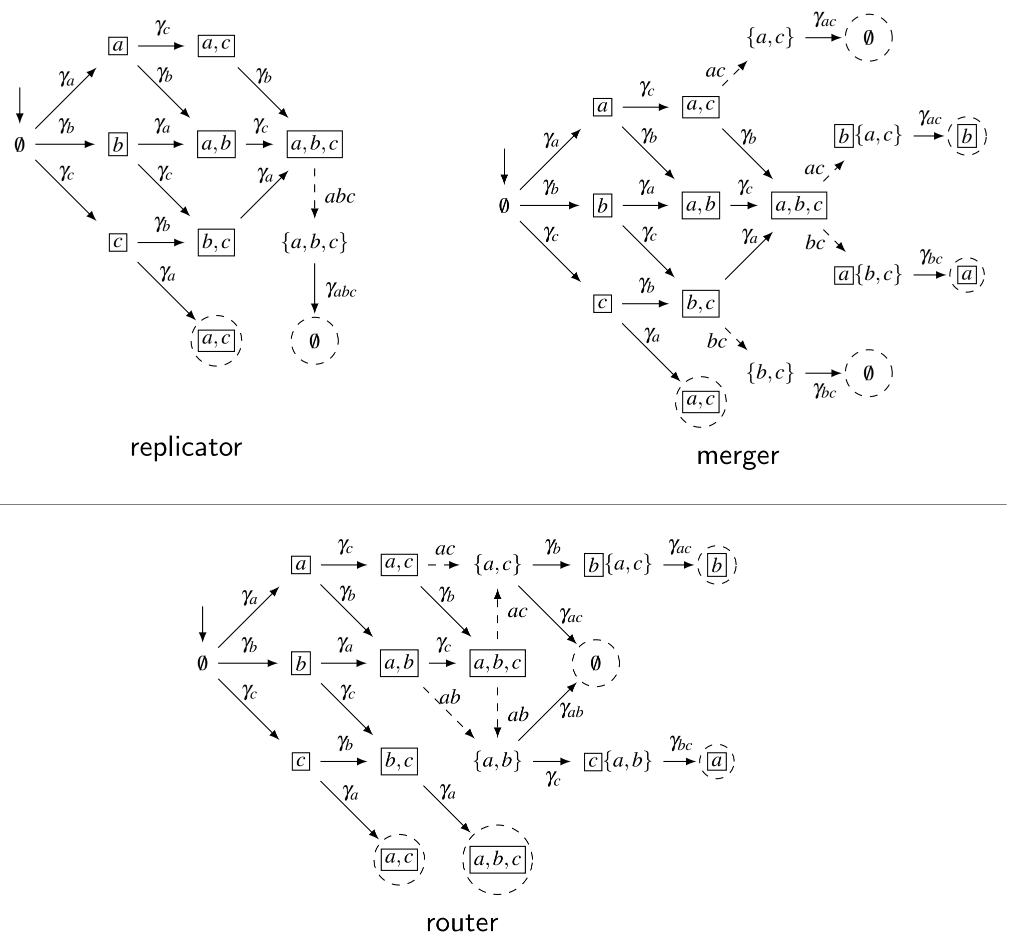
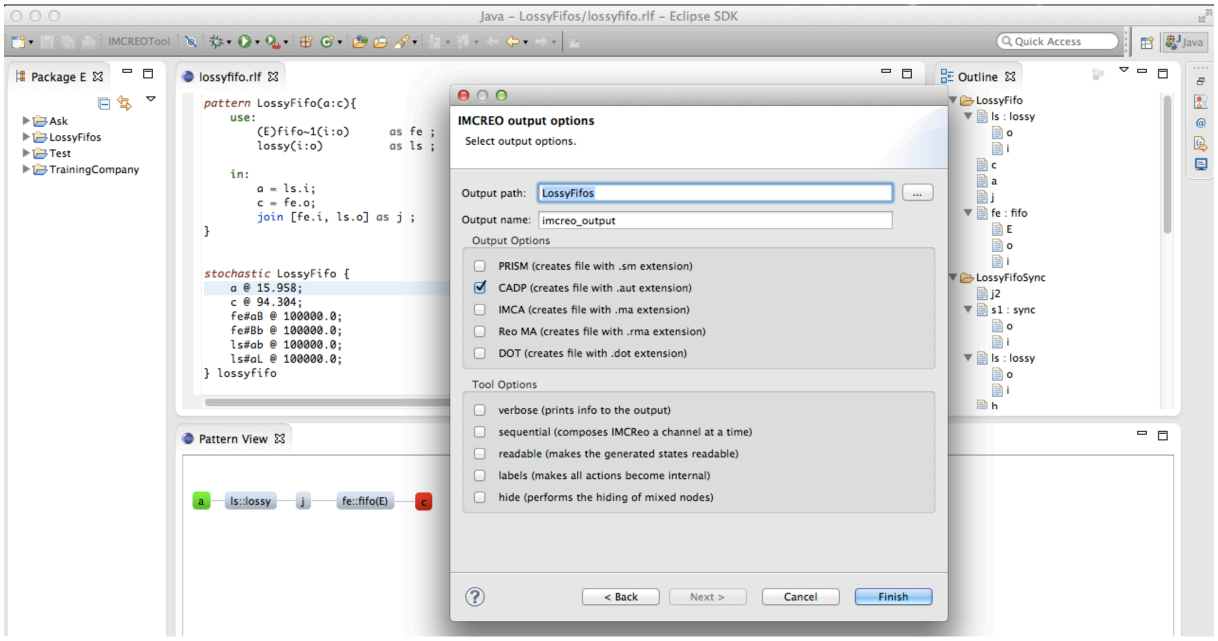


Figure 14: IMC$_{\texttt{Reo}}$ for the replicator, merger and router three-port channels.

CooPLa was recently adapted to Stochastic Reo, making it possible to model arrival rates and processing delays in several instances of a single pattern. In Figure 15, a stochastic instance lossyfifo of the LossyFifo pattern is presented. It associates arrival rates to the ports of the pattern, and processing delays to the flows in its channels (marked with an identifier preceded by a #).

The second element in the tool chain is an engine, referred to as IMCREOtools, which translates a Stochastic Reo connector to an IMC$_{\texttt{Reo}}$ model. It is based on two main components: the *generator* and the *prismer*. The *generator* takes the CooPLa description of a connector and processes it by composing a (randomly selected) initial channel with the subsequent ones. This creates an IMC$_{\texttt{Reo}}$ model that can be exported in several formats accepted by typical IMC tools, for example CADP [21] or IMCA [22]. The *prismer* component converts an IMC$_{\texttt{Reo}}$ model into a format compatible with the PRISM probabilistic model-checker [29].

IMCREOtools exists both as a command line tool and a plugin for the CooPLa editor (as shown in Figure 15). The plugin improves the tool usage by automating tasks that in the command line version require user intervention and acquaintance with specific tools (*e.g.* CADP).



```
pattern LossyFifo(a:c){
    use:
        (E)fifo~1(i:o)        as fe ;
        lossy(i:o)            as ls ;

    in:
        a = ls.i;
        c = fe.o;
        join [fe.i, ls.o] as j ;
}


stochastic LossyFifo {
    a @ 15.958;
    c @ 94.304;
    fe#aB @ 100000.0;
    fe#Bb @ 100000.0;
    ls#ab @ 100000.0;
    ls#aL @ 100000.0;
} lossyfifo
```

IMCREO output options

Select output options.

Output path: LossyFifos

Output name: imcreo_output

Output Options

- ☐ PRISM (creates file with .sm extension)
- ☑ CADP (creates file with .aut extension)
- ☐ IMCA (creates file with .ma extension)
- ☐ Reo MA (creates file with .rma extension)
- ☐ DOT (creates file with .dot extension)

Tool Options

- ☐ verbose (prints info to the output)
- ☐ sequential (composes IMCReo a channel at a time)
- ☐ readable (makes the generated states readable)
- ☐ labels (makes all actions become internal)
- ☐ hide (performs the hiding of mixed nodes)

Figure 15: The CooPLa development environment.

5.1.2 Quantitative analysis

For analysis the tool chain resorts to powerful external tools, namely IMCA, CADP or PRISM. The CADP tool plays an important role in it since it is able to perform both transient- and steady-state analysis. Moreover it may be used as an interface for other tools in the chain: for example, *bcg_io*, to convert aut models (a possible output format of IMCREOtools) into bcg models (the input format for CADP); *reductor* (along with *bcg_open*), to reduce IMC with τ-transitions; or *bcg_min*, to minimise IMC with respect to different bisimulation algorithms. The IMCA tool also allows for performance analysis with focus on reachability probabilities. It is not as powerful as CADP, but depending on the objectives of the analysis it may be a suitable alternative. Its input is a textual representation of a Markov automaton (ma) which IMCREOtools is able to export. The PRISM model checker does not accept IMC, so it is necessary to convert IMC$_{\mathsf{Reo}}$ models into continuous-time Markov chains (CTMC). This transformation starts by generating an IMC$_{\mathsf{Reo}}$ model where all interactive transitions are made internal. This model is then passed to the *reductor* CADP tool[2] for both minimisation with a *weak trace* equivalence relation and determinisation. In the end, the resultant CTMC model is serialised into an aut file that is converted by the *prismer* tool of IMCREOtools, into sm PRISM files. For stochastic analysis, PRISM provides a comprehensive set of facilities which largely extends the analysis scope.

The graph depicted in Figure 16 illustrates some possibilities of combining these tools for quantitative analysis of IMC$_{\mathsf{Reo}}$ models.

5.2 The ASK system

The ASK (*Access Society's Knowledge*) system is an industrial solution from the Dutch company Almende. It is a communication software that facilitates mediation between consumers and service pro-

[2]http://cadp.inria.fr/man/reductor.html

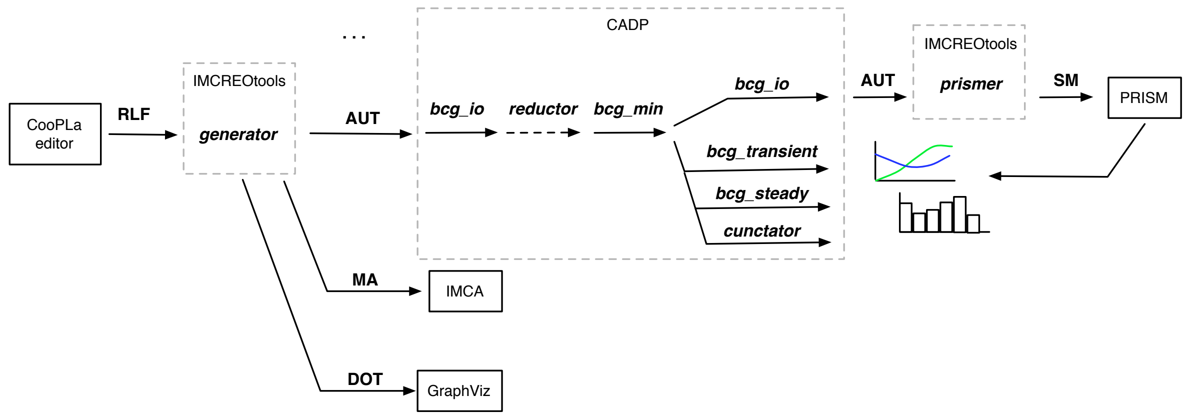


Figure 16: Tool chain for the analysis of $\texttt{IMC}_{\texttt{Reo}}$ models.

viders. The mediation is made by matching mechanisms which take into account the needs of the consumers and the profiles of the providers. Performing these matches and connecting the matched entities in an efficient and correct way is the main objective of this system.

The architecture of the ASK system is modular, based on components that, in turn, are built from smaller components communicating with each other. The higher level components are a web front-end, a database, and a contact engine. The front-end serves as the interface between the consumers/providers and the system; the database stores typical system data, along with specific data for the improvement of the matching algorithm. The contact engine consists of the following components: a Reception, which receives the requests and converts them into tasks to be processed to fulfil the request; a Matcher, which determines the best providers/consumers to attend a request; an Executer, which defines how the matched entities should communicate; a Resource Manager, which establishes the connection between the matched entities; and a Scheduler, which schedules requests taking into account time constraints of requests stored in the database.

The ASK's workflow considered in this case study starts with the issuing of requests by consumers (or triggered by the scheduler). These requests are sent to the Reception component, wherein a *HostessTask* (HT) subcomponent converts them into tasks which are enqueued into a Task-Queue (TQ). These tasks are later dequeued by a *HandleRequestTask* (HRT) subcomponent, which generates new requests to the Executer component. Within the Executer requests are also enqueued into an Execution-Queue (EQ) until a *HandleRequestExecution* (HRE) subcomponent is ready to take each of them and convert it into a connection between a service provider and the consumer.

Dealing with the high number of requests, imposes the concurrent execution of several instances of subcomponents (of the Reception and Executer components) in multiple threads. Moreover, since processing is not instantaneous, queues have to store incoming requests. Thus, analysing the system from the perspective of the resources needed (*e.g.*, number of threads running or memory used) is of paramount importance to deliver a cost-effective performance. $\texttt{IMC}_{\texttt{Reo}}$ provides the means to leverage such analysis, and the associated tools allow for investigating system quantitative issues, namely, performance, costs and resource requirements.

In [36, 33], the ASK system was thoroughly described, and part of it (the Reception component only) was first analysed for its performance and resources allocation, resorting to basilar `Stochastic Reo` tooling. In the sequel, we revisit parts of the problem addressed in [36, 33] within the approach proposed in this paper. Furthermore, we investigate correlations between memory usage and threads running to infer optimal performance in the Executer component.

5.2.1 Modelling: rates and delays

The case study focus on both the Reception and the Executer components. In particular, we investigate their interplay and reason about allocation of resources, such as memory blocks and running threads, to obtain a cost-effective performance of the overall system. Figure 17 shows the relevant part of the ASK system involving the Reception and the Executer components with their internal subcomponents. For simplicity, only one instance of subcomponents HT, HRT, and HRE is represented. Actually, this should be regarded as a set of instances, each one running on its own thread. Elements TQ and EQ represent queues that temporarily store tasks, as described above. The values over arrows represent time intervals (in seconds) suitably explained in Figure 17.

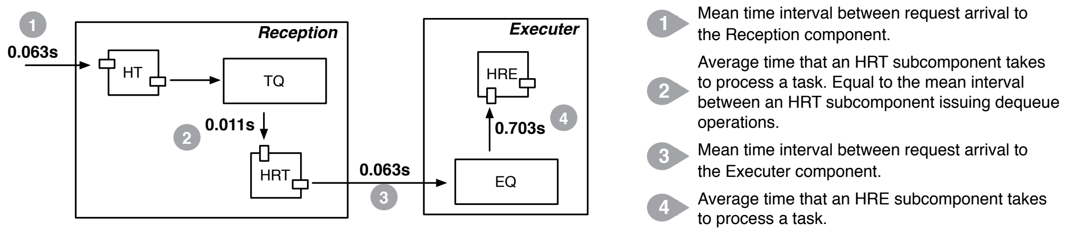

1
0.063s
Reception
HT
TQ
0.011s
2
HRT
0.063s
3
Executer
HRE
0.703s
4
EQ
1 Mean time interval between request arrival to the Reception component.
2 Average time that an HRT subcomponent takes to process a task. Equal to the mean interval between an HRT subcomponent issuing dequeue operations.
3 Mean time interval between request arrival to the Executer component.
4 Average time that an HRE subcomponent takes to process a task.

Figure 17: The ASK system model: Reception and Executer components.

Note that the data used for this case study was real, provided by the ASK team, as it is the case in [33, 34]. However, in certain aspects, the relevant values extracted differ from the values used in the latter. Moreover, delays of data flow between architectural elements are disregarded, as we assume them to be contemplated in the processing delays of the elements. As a consequence the concrete results will differ from [33, 34], precluding a detailed comparison, but not biasing the final conclusions.

Modelling in IMC$_{\texttt{Reo}}$ presupposes that the stochastic information follows an exponential distribution with some rate parameters. Without loss of generality, we assume that the values extracted for arrival rates and processing delays of components do refer to exponential distributions. Therefore, the rate parameters of the stochastic values in the ASK model are given as mi^{-1}, where mi refers to each mean time interval represented in Figure 17.

5.2.2 Analysis 1: memory issues

An important requirement of the ASK system is that it always runs smoothly, attending all incoming requests. However, the processing of these requests is not instantaneous and therefore must be handled asynchronously. Queues TQ and EQ support asynchrony, but are not unlimited and storage has an associated cost which is intended to be kept minimal.

In this context, we want to find out how much storage capacity will be necessary in order to make the system alive. Similarly to [36, 33] we assume that both TQ and EQ are unbounded. Such an unbounded queue is mimicked as a lossyfifo stochastic Reo connector. Then we can use IMC$_{\texttt{Reo}}$ and associated tools to check the amount of lost requests, and to derive a minimal value of storage capacity for queues. Figure 18 depicts the stochastic Reo connectors that abstract the coordination layer in the Reception and the Executer components. Notice that the processing delay of the HT subcomponent is negligible. Thus we can assume that the rate of requests arriving to TQ is equal to the rate of arrivals at HT. Notice how ∞ is used to represent a rate parameter so high that data flow delays can be safely ignored.

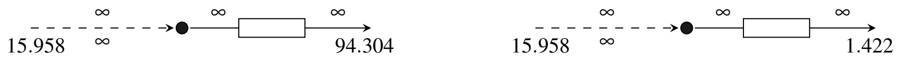
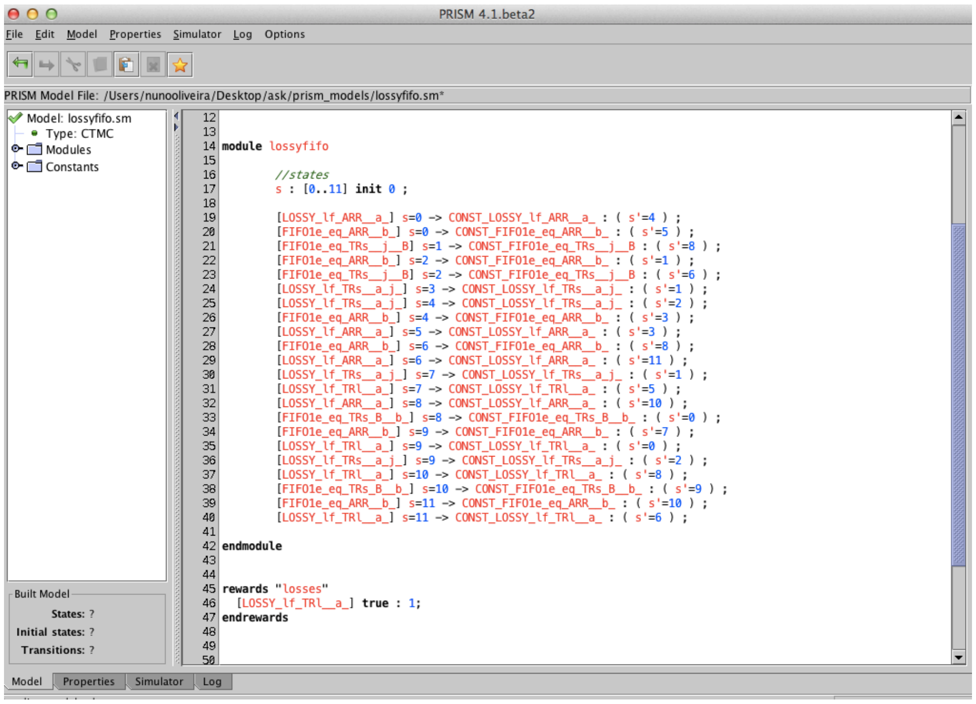


Figure 18: A lossyfifo modelling unbounded versions of TQ (left) and EQ (right).

These connectors were created with the `CooPLa` editor and then supplied to `IMCREOtools`. The resulting `IMC`$_{\texttt{Reo}}$ models have 19 states and 33 transitions, each one. After minimisation and conversion to a `CTMC` with `CADP`, this is reduced to 12 states and 22 transitions. Finally, the *prismer* tool of `IMCREOtools` builds the corresponding `PRISM` model (Figure 19). In `PRISM`, a reward structure assigning a unitary value to each loss is defined. Notice that *prismer* adds a `PRISM` action to each transition, which makes simpler the definition of such reward structures and increases the legibility of the model.


PRISM 4.1.beta2
File Edit Model Properties Simulator Log Options
PRISM Model File: /Users/nunooliveira/Desktop/ask/prism_models/lossyfifo.sm*
Model: lossyfifo.sm
Type: CTMC
Modules
Constants
Built Model
States: ?
Initial states: ?
Transitions: ?
Model Properties Simulator Log

```


module lossyfifo

        //states
        s : [0..11] init 0 ;

        [LOSSY_lf_ARR__a_] s=0 -> CONST_LOSSY_lf_ARR__a_ : ( s'=4 ) ;
        [FIFO1e_eq_ARR__b_] s=0 -> CONST_FIFO1e_eq_ARR__b_ : ( s'=5 ) ;
        [FIFO1e_eq_TRs__j__B] s=1 -> CONST_FIFO1e_eq_TRs__j__B : ( s'=8 ) ;
        [FIFO1e_eq_ARR__b_] s=2 -> CONST_FIFO1e_eq_ARR__b_ : ( s'=1 ) ;
        [FIFO1e_eq_TRs__j__B] s=2 -> CONST_FIFO1e_eq_TRs__j__B : ( s'=6 ) ;
        [LOSSY_lf_TRs__a_j_] s=3 -> CONST_LOSSY_lf_TRs__a_j_ : ( s'=1 ) ;
        [LOSSY_lf_TRs__a_j_] s=4 -> CONST_LOSSY_lf_TRs__a_j_ : ( s'=2 ) ;
        [FIFO1e_eq_ARR__b_] s=4 -> CONST_FIFO1e_eq_ARR__b_ : ( s'=3 ) ;
        [LOSSY_lf_ARR__a_] s=5 -> CONST_LOSSY_lf_ARR__a_ : ( s'=3 ) ;
        [FIFO1e_eq_ARR__b_] s=6 -> CONST_FIFO1e_eq_ARR__b_ : ( s'=8 ) ;
        [LOSSY_lf_ARR__a_] s=6 -> CONST_LOSSY_lf_ARR__a_ : ( s'=11 ) ;
        [LOSSY_lf_TRs__a_j_] s=7 -> CONST_LOSSY_lf_TRs__a_j_ : ( s'=1 ) ;
        [LOSSY_lf_TRl__a_] s=7 -> CONST_LOSSY_lf_TRl__a_ : ( s'=5 ) ;
        [LOSSY_lf_ARR__a_] s=8 -> CONST_LOSSY_lf_ARR__a_ : ( s'=10 ) ;
        [FIFO1e_eq_TRs_B__b_] s=8 -> CONST_FIFO1e_eq_TRs_B__b_ : ( s'=0 ) ;
        [FIFO1e_eq_ARR__b_] s=9 -> CONST_FIFO1e_eq_ARR__b_ : ( s'=7 ) ;
        [LOSSY_lf_TRl__a_] s=9 -> CONST_LOSSY_lf_TRl__a_ : ( s'=0 ) ;
        [LOSSY_lf_TRs__a_j_] s=9 -> CONST_LOSSY_lf_TRs__a_j_ : ( s'=2 ) ;
        [LOSSY_lf_TRl__a_] s=10 -> CONST_LOSSY_lf_TRl__a_ : ( s'=8 ) ;
        [FIFO1e_eq_TRs_B__b_] s=10 -> CONST_FIFO1e_eq_TRs_B__b_ : ( s'=9 ) ;
        [FIFO1e_eq_ARR__b_] s=11 -> CONST_FIFO1e_eq_ARR__b_ : ( s'=10 ) ;
        [LOSSY_lf_TRl__a_] s=11 -> CONST_LOSSY_lf_TRl__a_ : ( s'=6 ) ;

endmodule


rewards "losses"
  [LOSSY_lf_TRl__a_] true : 1;
endrewards



```

Figure 19: Derived `PRISM` model for a lossyfifo stochastic connector.

We may now discuss storage dimensioning by investigating the expected number of lost requests in the long-run, through the query ‘`R{"losses"}=?[S]`’. The expected loss is 0.4 requests per second for the TQ and 14.5 for the EQ (both results were computed in approximately 0.004s, which is a substantial gain when compared with the same process in [33]: 3.29s).

This fixes the optimal storage capacity for the queues in 2 for the TQ and 16 for the EQ. Notice that these numbers are based on the assumption that losses will occur when the buffer of the fifo channel is full. These storage limits will ensure that no requests are lost, and therefore the system will always accept new requests. Clearly, these results are only valid for average scenarios. Worst-case and best-case scenarios are not dealt with in this study.

5.2.3 Analysis 2: the Executer performance

Both the Reception and the Executer components rely on multithreading to haste the processing of incoming tasks. But is this really necessary? To find out we may compute the probability of a single instance of HRT and HRE actually initiating its execution per second.

Computing ‘`R{"run"}=?[ S ]/15.958`’ in `PRISM`, returns the probability of an instance to start running. Notice that the `"run"` is another reward structure associated to transitions that lead a request from the queue to the subcomponent instance (HRT or HRE in either model). It is found out that each instance of subcomponents HRT and HRE start executing with probability 97.6% and 8.9%, respectively. On the one hand, the need for multithreading on the Reception component is not a must, since a single instance takes care of the incoming requests quite well. On the other hand, the Executer component would benefit from it, given the low probability with which an instance of the HRE subcomponent starts running.

Resorting again to IMC$_{\texttt{Reo}}$, two different configurations of the Executer architecture were modelled to experiment with different uses of concurrency. Figure 20 shows the corresponding `Stochastic Reo` connectors. Both are based on the lossyfifo previously used, but equipped with a router connector to route requests from the EQ to each HRE instance.

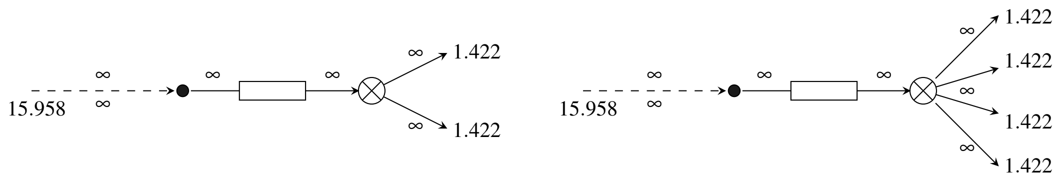


Figure 20: Concurrency alternatives: 2-HRE (left) and 4-HRE (right) versions.

The derived IMC$_{\texttt{Reo}}$ models of these connectors are quite large, even after minimisation. The 2-HRE version has 66 states and 155 transitions, and the 4 HRE version counts on 690 states and 1723 transitions. The reason is that a router connector with n output ports will be refactored into $n-1$ three-ports router channels (*c.f.*, Figure 13), and, in turn, each of these channels is modelled as an already complex IMC$_{\texttt{Reo}}$ chain (*c.f.*, Figure 14). However, these numbers place no problems for `PRISM`, as can be seen from the execution times presented in the last column of Table 1.

Table 1: Memory and execution analysis (per second) on 2-HRE and 4-HRE models.

Model	**Losses (requests)**	**Storage (requests)**	**Execution (%)**	`PRISM` **Time (s)**
2-HRE	13.15	15	17.55	0.09
4-HRE	10.47	12	34.38	4.218
HREx2	13.19	15	17.35	0,004
HREx4	10.75	12	32.59	0.004

Analysing these models with respect to storage needs and the execution probability properties already discussed, leads to the values shown in Table 1 (first two rows). Not surprisingly, the more instances set in concurrent threads, the less losses will occur, and therefore, the lower the need for storage capacity becomes. For the execution probability, it is observed a linear growth by a factor close to 2. From here, it is predictable that 12 would be an optimal number for concurrent HRE instances in order to maximise the probability with which they start executing at each second.

We played further with these models to understand what would be the difference between adding concurrency or increasing individual performance (by diminishing the processing delay in individual components) by the same scalar factor. The results relative to the latter alternative are shown in the last two rows of table 1. While for low values of this factor the differences are not that significative, for higher values they grow exponentially. In general improving the processing time of a single instance, is less efficient than increasing concurrency.

6 Related work

6.1 Semantic models for `Stochastic Reo`

Many semantic models for `Reo` have been proposed in the last few years [27], usually capturing different aspects of the coordination language. Constraint automata [10], for example, makes explicit the constraints on data when flowing through nodes that synchronously fire in each step of execution. Intensional automata [18], on the other hand, focus on both the internal and the external state of the connector. The former concerns its buffering capacities; the latter deals with the arrivals of `I/O` requests to the connector boundary nodes. This model, differently from constraint automata, is able to capture context-dependency. The same is true of `Reo` automata [15, 16], which do it in a more natural way using guards to assert conditions on the presence or absence of requests at the connector ports, and define the firing of `I/O` actions.

Most of these automata-based semantic models for `Reo` have been extended to support the stochastic behaviour. For example, continuous-time constraint automata [11] extend constraint automata by incorporating continuous-time. Therefore, they introduce a new type of transitions that mimic continuous-time Markov chains, resembling the notion of `IMC` and, therefore, the $\mathtt{IMC}_{\mathtt{Reo}}$ model proposed in this paper. The model inherits relevant properties from constraint automata, namely compositionality, but fails to capture context-dependency.

Quantitative intensional automata [3], spring from intensional automata by adding extra information to transitions: data constraints, as in constraint automata, and quantitative information to model delays and arrival rates of interaction actions. This model correctly captures context dependency, but it suffers from state explosion even when representing simple connectors which restricts scalability. Moreoever it is not a compositional model, because of its unstructured composition operator.

`Stochastic Reo` automata [35, 36], extend `Reo` automata by incorporating processing delays into transitions, and arrival rates of `I/O` requests into the nodes of the `Reo` connector. The resulting models are compact, capture context-dependency and compositionality is inherited from `Reo` automata. The practical use of stochastic `Reo` automata is, however, constrained by the lack of tool support. In an attempt to bridge this gap, in [36] partial translations were provided to `CTMC` and `IMC`, so that tool support from these standard models could be used. However, the translations were shown not to be compositional, which forces the user to recalculate the whole model every time a tiny change occurs in the connector.

If our research has `Reo` like coordination as its starting point, a reference is in order to other approaches that inspect stochastic properties of composed systems based on coordination models. Stochastic BIP (SBIP) [13] is one of the most recent and acclaimed examples. SBIP is a stochastic extension of the BIP coordination model [12]. It allows for the specification of stochastic aspects of atomic components in BIP and introduces a stochastic semantics based on transition systems, modelling both stochastic and non-deterministic behaviour. SBIP is supported by suitable tools and algorithms that cater for statistical model checking of quantitative and qualitative properties expressed in (probabilistic) bounded linear temporal logic. A major feature of SBIP is the possibility of specifying systems with non-deterministic behaviour. However it must be determinised, using a notion of priority scheduling in BIP, for analysis with the statistical model checker. Our approach is likewise. $\mathtt{IMC}_{\mathtt{Reo}}$ also allows for non-determinism but, for instance, its analysis in PRISM requires a purely stochastic model. Such a conversion is made by determinising IMC$_{\texttt{Reo}}$ models with CADP. Differently from IMC$_{\texttt{Reo}}$, SBIP is not restricted to the exponential distribution.

6.2 Other stochastic models

IMC were used in this paper as a semantic basis for Stochastic Reo. However, a plethora of semantic models dealing with random processes exists in the literature. In the domain of Markovian models, CTMC lack of compositionality, but Markov automata [19] subsume IMC and provides flexible ways to model probabilistic behaviour. This could replace IMC in the approach proposed here by replacing interactive transitions by probabilistic ones with some distribution associated. However, Stochastic Reo does not take this into account (interactions occur with constant probability of 1) and therefore IMC do qualify as a more concise alternative.

Stochastic Petri nets [32, 30] associate exponential distributions to transition firing, which allows the derivation of CTMC from the graph-based specifications of Petri nets. However, it is not a compositional model and does not take into account the global synchrony of events that is required in Reo. Differently, stochastic process algebra [26, 14, 17] is a compositional formalism that incorporates both process algebra and stochastic processes in one specification framework, as much as IMC do. The main drawbacks for their use on this work is the size of the transition systems generated by their operational semantics as well as the lack of specific tool support.

Queueing theory [23] also plays an important role in the stochastic modelling of systems. In fact, components can be reduced to queues and a system to a network of queues. A queueing network can thus be regarded as a graph whose nodes are queueing stations (with some service rate) and edges are connections between queues (with some probability associated). The performance analysis of such networks is also based on the theory of Markov chains (mainly on CTMC). Although Reo channels can be modelled as queues and a connector modelled as a network, there are several Reo properties that are not covered by such a theory, at least in a direct way. A typical example is context-awareness.

A similar alternative are stochastic automata networks [40, 20], a model that became popular in the late nineties to formalise parallel and distributed systems. Each component is modelled as a stochastic automaton that interact with the others. It mitigates the state space explosion problem of Markovian models as they are represented by small matrices, and relevant properties of the system may be inspected without the need for computing a global matrix. But a requirement of these systems is that they operate nearly independently and very rarely synchronise. Reo is otherwise; it requires a global notion of synchronisation.

In this context, IMCs seem to provide a suitable, yet simple alternative as a semantic basis for Stochastic Reo, with suitable tool support.

7 Conclusions

The Stochastic Reo semantics proposed in this paper, and directly based in Interactive Markov Chains, aims at providing a direct stochastic semantics to this coordination framework. In comparison with the approach proposed in [36], the basic insight is that the absence of an intermediate automata model avoids extra translation steps; and makes possible to directly inheriting compositionality from classic IMC. Moreover, IMC is a well-studied formalism with several tools and a mature theory. Therefore, typical measures obtained from IMC analysis become available for the study of stochastic Reo circuits. A tool — IMCREOtools[3] — was developed to translate Reo connectors into IMC$_{\texttt{Reo}}$. Its output is compatible

[3]http://reo.project.cwi.nl/reo/wiki/ImcReo

with several well-known tools for IMC (e.g. CADP [21], PRISM [29], and IMCA [22]), leading to a tool chain for qualitative and quantitative analysis of coordination models. An important point to note is that these analysis tools do scale up (see *e.g.*[31] in which chains with millions of states are analysed).

Avoiding exponentially growth in the size of the generated $\mathtt{IMC}_{\mathtt{Reo}}$ chains remains, nonetheless, a relevant issue. As future work we intend to explore recent work on Reo [28], which introduces a notion of connector partitions for distributing connector schemes over multiple machines.

Other directions for future work include relaxing the Reo assumption on mixed nodes as *self-pumping* stations, which allow for data to be read and written with no processing delay. In practice this assumption is unrealistic: I/O operations take time and, therefore, may interfere with QoS values. On the technological side, we intend to achieve a full integration of the proposed tool chain in ECT [4], a plugin-based environment to model and analyse Reo coordination under Eclipse.

Acknowledgments. We thank Farhad Arbab for several suggestions and comments. This work is funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT, the Portuguese Foundation for Science and Technology, within project FCOMP-01-0124-FEDER-020537. The first author is also supported by an Individual Doctoral Grant from FCT, with reference SFRH/BD/71475/2010.

References

[1] Farhad Arbab. Abstract behavior types: A foundation model for components and their composition. In *Formal Methods for Components and Objects*, volume 2852 of *LNCS*, chapter 2, pages 33–70. Springer-Verlag, 2003.

[2] Farhad Arbab. Reo: a channel-based coordination model for component composition. *Mathematical Structures in Computer Science*, 14(3):329–366, June 2004.

[3] Farhad Arbab, Tom Chothia, Rob van der Mei, Sun Meng, Young-Joo Moon, and Chrétien Verhoef. From coordination to stochastic models of QoS. In *Coordination Models and Languages*, volume 5521 of *LNCS*, chapter 14, pages 268–287. Springer-Verlag, 2009.

[4] Farhad Arbab, Christian Krause, Ziyan Maraikar, Young-Joo Moon, and José Proença. Modeling, testing and executing Reo connectors with the eclipse coordination tools. In *Proc. of the 5th International Workshop on Formal Aspects of Component Software (FACS 2008), Malaga, Spain*, September 2008.

[5] Farhad Arbab and Farhad Mavaddat. Coordination through channel composition. In *Coordination Models and Languages*, volume 2315 of *LNCS*, chapter 6, pages 275–297. Springer-Verlag, March 2002.

[6] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert Brayton. Model-checking continuous-time markov chains. *ACM Transactions in Computer Logic*, 1:162–170, July 2000.

[7] J. C. M. Baeten. A brief history of process algebra. *ENTCS*, 335(2-3):131–146, May 2005.

[8] Christel Baier, Tobias Blechmann, Joachim Klein, and Sascha Klüppelholz. A uniform framework for modeling and verifying components and connectors. In *Coordination Models and Languages*, volume 5521 of *LNCS*, chapter 13, pages 247–267. Springer-Verlag, 2009.

[9] Christel Baier, Boudewijn Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-Checking algorithms for Continuous-Time markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.

[10] Christel Baier, Marjan Sirjani, Farhad Arbab, and Jan Rutten. Modeling component connectors in Reo by constraint automata. *Science of Computer Programming*, 61(2):75–113, 2006.

[11] Christel Baier and Verena Wolf. Stochastic reasoning about channel-based component connectors. In *Proc. of the 8th international conference on Coordination Models and Languages (COORDINATION'06), Bologna, Italy, LNCS*, volume 4038, pages 1–15. Springer-verlag, June 2006.

[12] Ananda Basu, Saddek Bensalem, Marius Bozga, Jacques Combaz, Mohamad Jaber, Thanh-Hung Nguyen, and Joseph Sifakis. Rigorous Component-Based system design using the BIP framework. *Software, IEEE*, 28(3):41–48, May 2011.

[13] Saddek Bensalem, Marius Bozga, Benoit Delahaye, Cyrille Jegourel, Axel Legay, and Ayoub Nouri. Statistical model checking QoS properties of systems with SBIP. In *Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change*, volume 7609 of *LNCS*, pages 327–341. Springer-Verlag, 2012.

[14] Marco. Bernardo and Roberto Gorrieri. Extended markovian process algebra. In *CONCUR'96: Concurrency Theory*, volume 1119 of *LNCS*, pages 315–330. Springer-Verlag, 1996.

[15] Marcello Bonsangue, Dave Clarke, and Alexandra Silva. Automata for Context-Dependent connectors. In *Proc. of the 11th International Conference on Coordination Models and Languages (COORDINATION'09), Lisbon Portugal, LNCS*, volume 5521 of *LNCS*, pages 184–203. Springer, June 2009.

[16] Marcello M. Bonsangue, Dave Clarke, and Alexandra Silva. A model of context-dependent component connectors. *Science of Computer Programming*, 77(6):685–706, June 2012.

[17] Allan Clark, Stephen Gilmore, Jane Hillston, and Mirco Tribastone. Stochastic process algebras. In *Formal Methods for Performance Evaluation*, volume 4486 of *LNCS*, pages 132–179. Springer-Verlag, 2007.

[18] David Costa. *Formal Models for Component Connectors*. PhD thesis, Vrije University, October 2010.

[19] Yuxin Deng and Matthew Hennessy. On the semantics of markov automata. *Information and Computation*, 222:139–168, January 2013.

[20] Paulo Fernandes, Brigette Plateau, and William J. Stewart. Efficient descriptor-vector multiplications in stochastic automata networks. *J. ACM*, 45(3):381–414, May 1998.

[21] Hubert Garavel, Frédéric Lang, Radu Mateescu, and Wendelin Serwe. CADP 2011: a toolbox for the construction and analysis of distributed processes. *International Journal on Software Tools for Technology Transfer*, pages 1–19, 2012.

[22] Denis Guck, Tingting Han, Joost-Pieter Katoen, and Martin R. Neuhäußer. Quantitative timed analysis of interactive markov chains. In *NASA Formal Methods*, volume 7226 of *LNCS*, pages 8–23. Springer-Verlag, 2012.

[23] Moshe Haviv. *Queues: A Course in Queueing Theory*. Springer-Verlag, 2013 edition, June 2013.

[24] Holger Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*, volume 2428 of *LNCS*. Springer-Verlag, 2002.

[25] Holger Hermanns and Joost-Pieter Katoen. The how and why of Interactive Markov chains. In *Proc. of the 8th international conference on Formal methods for components and objects (FMCO'09), Eindhoven, The Netherlands, LNCS*, volume 6286, pages 311–337. Springer-Verlag, November 2010.

[26] Jane Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[27] Sung-Shik Jongmans and Farhad Arbab. Overview of thirty semantic formalisms for Reo. *Scientific Annals of Computer Science*, 22(1):201–251, 2012.

[28] Sung-Shik T. Q. Jongmans, Francesco Santini, and Farhad Arbab. Partially-Distributed coordination with reo. In *Proc. of the 22nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP'2014), Turin, Italy*, pages 697–706. IEEE, February 2014.

[29] Marta Kwiatkowska, Gethin Norman, and David Parker. A framework for verification of software with time and probabilities. In *Proc. of the 8th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'10), Klosterneuburg, Austria, LNCS*, volume 6246, pages 25–45. Springer-Verlag, September 2010.

[30] Marco A. Marsan. Stochastic petri nets: An elementary introduction. In *Advances in Petri Nets 1989*, volume 424 of *LNCS*, pages 1–29. Springer-Verlag, 1990.

[31] Radu Mateescu and Wendelin Serwe. Model checking and performance evaluation with CADP illustrated on shared-memory mutual exclusion protocols. *Science of Computer Programming*, 78(7):843–861, 2013.

[32] Michael K. Molloy. *On the Integration of Delay and Throughput Measures in Distributed Processing Models*. PhD thesis, University of California, 1981.

[33] Young-Joo Moon. *Stochastic Models for Quality of Service of Component Connectors*. PhD thesis, Universiteit Leiden, October 2011.

[34] Young-Joo Moon, Farhad Arbab, Alexandra Silva, A. Stam, and Chrétien Verhoef. Stochastic reo: a case study. In *Proc. of the 5th International Workshop on Harnessing Theories for Tool Support in Software (TTSS*

'11), Oslo, Norway, 2011.

[35] Young-Joo Moon, Alexandra Silva, Christian Krause, and Farhad Arbab. A compositional semantics for stochastic Reo connectors. In *Proc. of the 9th International Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA'10) Paris, France*, pages 93–107, September 2010.

[36] Young-Joo Moon, Alexandra Silva, Christian Krause, and Farhad Arbab. A compositional model to reason about end-to-end QoS in stochastic Reo connectors. *Science of Computer Programming*, December 2011.

[37] Nuno Oliveira and Luís S. Barbosa. On the reconfiguration of software connectors. In *Proc. of the 28th Annual ACM Symposium on Applied Computing (SAC '13), Coimbra, Portugal*, volume 2, pages 1885–1892. ACM, March 2013.

[38] Nuno Oliveira and Luís S. Barbosa. Reconfiguration mechanisms for service coordination. In *Web Services and Formal Methods*, volume 7843 of *LNCS*, pages 134–149. Springer-Verlag, 2013.

[39] Nuno Oliveira, Alexandra Silva, and Luís S. Barbosa. Quantitative analysis of Reo-based service coordination. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC '14), Gyeongju, Republic of Korea*, volume 2, pages 1247–1254. ACM, March 2014.

[40] William J. Stewart, Karim Atif, and Brigette Plateau. The numerical solution of stochastic automata networks. *European Journal of Operational Research*, 86(3):503–525, November 1995.