Interactive Markov Chains Meet Stochastic Reo

Nuno Oliveira[1] and Alexandra Silva[2,1] and Luís S. Barbosa[1]

[1] HASLab/INESC TEC & Universidade do Minho
`{nunooliveira,lsb}@di.uminho.pt`
[2] Radboud University Nijmegen and Centrum Wiskunde & Informatica
`alexandra@cs.ru.nl`

Abstract. `QoS` analysis of composed software systems is an active research area, with the goal of evaluating and improving performance and resource allocation in component-based applications.

The study of composed software and its properties is in itself a great challenge and along the years many approaches have been proposed. 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 effective modeling and verification.

Stochastic `Reo` offers constructs for component and service coordination and provides means for specification of stochastic values for the channels (e.g., arrival and processing rates). Interactive Markov chains (`IMC`) were proposed as a stochastic, compositional, extension of classical models of concurrency, in which the classical and stochastic features are treated in an harmonious manner.

In this paper, we show how `IMC` can be effectively used to serve as semantic model for Stochastic `Reo`. Treating `IMC` as a first-call semantic 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. All the tool support already existing for `IMC` is made available, without significant effort, to verify and reason about `Reo` connectors.

1 Introduction

Component-based software engineering and service-oriented computing aim at the development of reusable software components and/or services as building blocks that can be composed to build different applications. The quest in this area is to ease the analysis of complex software components, by providing compositional models: properties of the composed system can be derived from the properties of its building blocks and the composing *glue*. Some approaches to software composition use textual glue code [28,18,31], usually in a scripting language, whereas others offer a more visual approach, where 'channels' or 'connectors' are used to compose components into a system [12,21,2,17].

Channel based-languages play a prominent role in the world of software composition. One of such languages is Reo [2,3], which offers a model of component and service coordination, wherein complex connectors are constructed by composing various types of primitive channels. Stochastic Reo [29] is an extension of Reo which allows for the specification of stochastic values for the channels (e.g., arrival and processing rates). Having stochastic values enables QoS analysis of composed software (intensive) systems, which has become popular in the last few years, with the goal of evaluating and improving performance and resource allocation in service-oriented applications.

There exist many semantic models in the literature for Reo [10,15,13,16,25,14], many of which fail to capture certain important features, such as the so-called context-dependency, which is a desired feature, characterised by behaviours which depend upon both the presence and absence of I/O requests on the boundary ports of the connector. For the stochastic extension of Reo, there are three main models: Continuous-Time Constraint Automata [11], stochastic intensional automata [4], and stochastic Reo automata [29,30]. The first model fails to capture context-dependency, problem inherited from constraint automata. The second model correctly captures context dependency, but it suffers from many drawbacks. For one, the number of states in the automata representing even simple connectors is large, which restricts immensely its applicability in real case studies. More worryingly, it is not a compositional model, because of the ad-hoc and contrived composition operator. The third model was proposed as a solution for both these drawbacks: the models are more compact and compositionally is inherited from Reo automata. However, the applicability of the latter model is also constrained by the lack of tool support. In an attempt to bridge the gap, in [30], partial translations were provided to CTMC and IMC, in the hope that then tool support from these standard models would be available for Reo. Unfortunately, the translations were shown to not be compositional, which results in recalculating the whole model every time a tiny change occurs in the automaton model, which has the obvious disadvantages and again compromises the applicability. Furthermore, the composition operator for IMC was shown to not be suitable for many Reo models.

CTCA - fails to capture context dependeny

SIA - state explosion for even simple connectors; not compositional

SRA - lack of tool support

S Reo --> SRA --> CTMC

This paper takes a different approach. Instead of having an intermediate automata model, we propose IMC as a semantic model for Stochastic Reo. This offers several challenges, in order to correctly capture the expressivity of Reo connectors and the composition of connectors. We show that the obtained model has many desired properties, including the important context dependency feature and compositionality, which enables a powerful analysis of complex systems. Tools like CADP [19] or IMCA [20] provide interesting and powerful means to analyse and model check IMC. The modelling of stochastic Reo with IMC enables, without significant effort, the use of these tools and associated techniques to reasoning about qualitative and quantitative aspects of Reo.

IMC for Reo
- **captures Context Dependency**
- **Compositional**
- **Tool Support (CADP + IMCA)**

S Reo -> IMC

Organisation The paper is organised as follows. In Section 2, we recall basic definitions of IMC and stochastic Reo. In Section 3, we define the IMC corresponding to basic Reo channels and discuss the adaptations that need to be done to the classical IMC model in order to provide a correct semantics for Reo. Section 4 contains the various operations needed to compose Reo channels and a compositionally result. We present concluding remarks and directions for future work in Section 5.

2 Background

In this section, we introduce basic material on Interactive Markov Chains, the coordination language Reo, and its stochastic variant.

2.1 Interactive Markov Chains

IMC - model for performance evaluation for distributed Systems

Combination of:
- Quantitative Modeling
- Process Algebra

Interactive Markov Chains (IMC) [22,23] were proposed as a model for performance evaluation of distributed systems. The approach combines systems quantitative modelling, based on continuous-time Markov chains (CTMC) [9,7], and process algebra [27,8], to ensure compositionally.

Interactive - capture the system interaction with the environment;

Markovian - model a random delay in the system evolution, governed by a negative exponential distribution with parameter \gamma

An IMC exhibits two sorts of transitions: *interactive* and *Markovian*. The former capture the system's interaction with its environment, and their occurrence is assumed not to be time consuming, once externally triggered. τ-labelled transitions abstract, as usual, unobservable activity. Since they do not interact with the environment, they are assumed to take place immediately, taking precedence over Markovian transitions. The latter 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 extends smoothly classical labelled transition systems, bringing to scene continuous time and specifying the delay probability for a state change.

Definition 1 (Interactive Markov Chain [23]). *An IMC is a tuple* $I = (S, Act, \dashrightarrow, \longrightarrow, s)$*, where*

- S *is a nonempty set of states.*
- Act *is set of actions.*
- $\dashrightarrow \subseteq S \times Act \times S$ *is the set of Interactive transitions.*
- $\longrightarrow \subseteq S \times \mathbb{R}^+ \times S$ *is the set of Markovian transitions.*
- $s \in S$ *is the initial state of the chain.*

Interactive transitions (s, a, s') are usually denoted by $s \overset{a}{\dashrightarrow} s'$ and represent a change in the system from state s to state s' through an external action a that may be executed either immediately or blocked until the environment triggers it. On the other hand, Markovian transitions (s, γ, s') are denoted by $s \xrightarrow{\gamma} s'$ and represent a transition from state s to state s' within t time units with a probability of $1 - e^{-\gamma . t}$.

Internal interactive transitions (τ-transitions) play an important role on IMC. Since they do not interact with the environment, no execution time is associated to them. Therefore, an internal transition always precedes any Markovian one specified for the same state (known as *unstable state*), because the probability

Maximal Progress Assumption:
- any tau-transition preceeds a markovian transition, because there is no prbability to transit from a state to another in 0 time units

to execute such transition within 0 time units is always null: $1-e^{-\gamma\cdot 0} = 1-e^{0} = 1-1 = 0$. This fact is known as the *maximal progress assumption*. Note that this only concerns Markovian transitions; interactive transitions may as well execute immediately.

2.2 `Reo` and Stochastic `Reo`

`Reo` [6,1,2] is a channel-based model for the exogenous coordination of components in the context of component-based software. A channel is a, normally, directed communication mean 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). Channels are composable to define more complex coordination structures referred to as connectors. Composition of channels is made on their ends, which form the nodes of 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 channel ends. The first two types may also be referred as the ports of the channel or connector. A channel is synchronous when it delays the operations at each of its ends so that they can succeed simultaneously. Otherwise a channel is asynchronous, exhibiting memory capabilities or the possibility of specifying an ordering policy for content delivering. A channel may also be lossy when it delivers some values but loses others depending on a specified policy. Fig. 1 recalls the basic channels used in `Reo`.

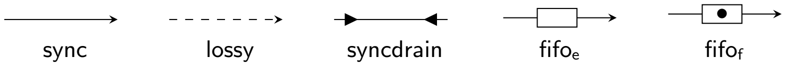

Fig. 1. Primitive Reo channels.

The sync channel transmits data from one end to another whenever there is a request at both ends synchronously, otherwise one request shall 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 `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.

S Reo = Reo + stochastic values on:
- ends: I/O arrival rates
- channels: processing delays

Stochastic `Reo` [4,29] extends `Reo` by modelling coordination with a quantitative perspective. Non-negative real (stochastic) values are added both to channels and to their ends to represent, respectively, *processing delay rates* and `I/O` *arrival rates*. The former rate models the time needed for the channel to process data from one point to another, where point may be seen as an 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. The latter models the time between consecutive arrivals of external `I/O` requests to channel ports.

Figure 2 shows the basic channels of stochastic `Reo`. In essence, they are the normal `Reo` channels but annotated with stochastic rates. Channel ends names are usually omitted because they can be inferred from the rate. Stochastic `Reo` is

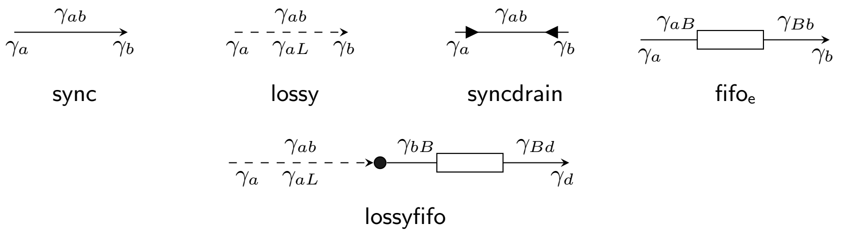

Fig. 2. Primitive stochastic Reo channels.

still composable. Each composed channel retains its processing delay rate. The request arrival rates, however, are only preserved for the ports of the connector. Since mixed nodes are internal (hidden from the exterior) the arrival request rates of the constituent nodes are ignored, meaning that they are always ready to read/write data from/to the channels. The lossyfifo connector in Figure 2 precisely depicts this situation.

No requests arrive to mixed nodes

3 Interactive Markov Chains for Stochastic Reo

This section discusses the formalisation of a semantic model for stochastic `Reo` as an instance of `IMC`. In order to capture `Reo`'s behaviour we will use an enriched set of labels for the interactive transitions and also a composed state space. Then, composition is defined building on the usual parallel composition for `IMC` and eliminating transitions which are not *`Reo`-like* via a synchronisation operator. This two-step composition is very much in the same spirit as the one defined for Reo automata [14].

3.1 The Semantic Model

Before introducing our proposal for an `IMC` model for stochastic `Reo`, referred to as $\texttt{IMC}_{\texttt{Reo}}$ 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 connector's possible behaviour, *i.e.*, data arrivals and data flowing through the ports. A set of node/port names, $\mathcal{N}$, and a set of state names, $\mathcal{Q}$, are assumed. Thus the state of a `Reo` connector comprises three components – (R, T, Q) – where

States: capture the connector behaviour
- data arrivals
- data flowing

- $R, T \in \mathcal{P}(\mathcal{N})$ denote the sets of ports with, respectively, pending requests and data being transmitted. Naturally, the empty set, $\emptyset$, represents absence of requests and transmissions.
- $Q \in \mathcal{P}(\mathcal{Q})$ is a set of internal state identifiers, which allows for distinguishing control states in state-based connectors, *i.e.*, connectors which admit different internal configurations, such as, for instance, a buffer. For the latter, for example, one must be able to distinguish between states corresponding to an empty and a full buffer. Thus, $\mathcal{Q}$ is taken as $\mathcal{Q} = \{\mathsf{empty}, \mathsf{full}\}$.

In the context of IMC modelling of (stochastic) Reo, Markovian transitions will be labelled by $r \in \mathbb{R}^+$, representing the delays according to rate r. Interactive transitions will be labelled by a set of ports F (corresponding to the observable actions) that fire and allow data to flow through them. Taking sets of actions to label transitions is crucial to correctly capture Reo semantics. Actually, ports firing synchronously to enable data flow, are the rule rather than the exception in Reo.

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

Definition 2 ($\mathtt{IMC}_{\mathtt{Reo}}$)**.** *An $\mathtt{IMC}_{\mathtt{Reo}}$ is a tuple $(S, Act, \dashrightarrow, \longrightarrow, s)$, where*

- $S \subseteq \mathcal{P}(\mathcal{N}) \times \mathcal{P}(\mathcal{N}) \times \mathcal{Q}$ *is a nonempty set of states;*
- $Act \in \mathcal{P}(\mathcal{N})$ *is a set of actions;*
- $\dashrightarrow \subseteq S \times Act \times S$ *is a set of Interactive transitions;*
- $\longrightarrow \subseteq S \times \mathbb{R}^+ \times S$ *is a set of Markovian transitions;*
- $s \in S$ *is the initial state.*

States of the form $(R, \emptyset, Q)$ are referred to as *request* states and are represented as $\boxed{R}_Q$; states of the form $(\emptyset, T, Q)$ are referred to as *transmission* states and are represented as $\{T\}_q$ or $\{t_T\}_Q$; states of the form (R, T, Q) are called as *mixed* states and are represented 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.

For simplicity, Markovian transitions (s, γ, s') are denoted by $s \xrightarrow{\gamma} s'$, and Interactive transitions $(s, \{a_1, a_2, ...\}, s')$ by $s \overset{a_1 a_2 \dots}{\dashrightarrow} s'$. An empty set of actions models the internal transition τ. To avoid graphical overlap of transitions, a dashed circle is used to refer to an already represented state.

Figure 3 depicts the $\mathtt{IMC}_{\mathtt{Reo}}$ for the basic stochastic Reo channels. For instance, the $\mathtt{IMC}_{\mathtt{Reo}}$ of a stochastic sync channel is interpreted as follows: initially, no requests are pending neither in port a nor in port b. At a rate of γ_a (resp. γ_b) a request arrives to port a (resp. port b). At that moment, the channel *blocks* until a request arrives to the other port at rate γ_b (resp. γ_a). 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 on a rate of γ_{ab}.

4 New connectors from old

This section contains the basic result in the paper: that IMC semantics for Stochastic Reo is compositional. Our starting point is the parallel composition of IMC [22], suitably tuned to deal with transitions labelled by *sets* of actions. From this, we get compositionality, as inherited from IMC, for free. However, parallel composition gives rise to a number of transitions which are not compatible with

IMC is compositional

IMC parallel composition is the starting point… however tuned to deal with set of actions

Parallel Composition => transitions not compatible with Reo Semantics

Synchrinising step

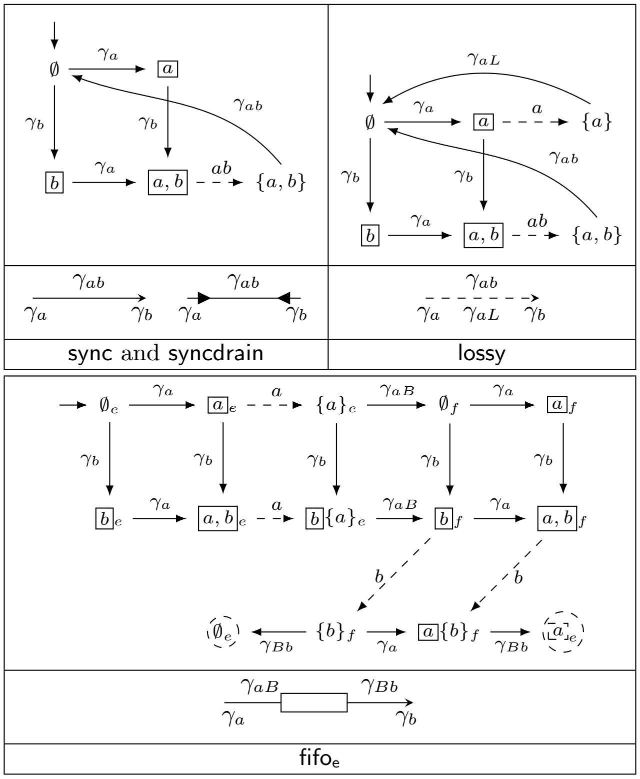

Fig. 3. `IMC` for basic stochastic `Reo` channels

the expected behaviour for `Reo` connectors, as discussed below. Thus, we further define a synchronisation operator which eliminates such transitions. It is shown that this synchronisation operator still preserves compositionality.

4.1 Parallel composition of connectors

Let us first recall the usual definition of `IMC` parallel composition, adapted to `IMC`$_{\texttt{Reo}}$ by explicitly dealing with sets of actions.

Definition 3 (Parallel Composition). *Let $I = (S_I, Act_I, \dashrightarrow_I, \longrightarrow_I, i)$ and $J = (S_J, Act_J, \dashrightarrow_J, \longrightarrow_J, j)$ be two `IMC`$_{\texttt{Reo}}$. The parallel composition of I and J with respect to a set of actions M is defined as*

$$I \,||_M\, J = (S, Act, \dashrightarrow, \longrightarrow, (i,j))$$

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

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 some $j \in S_J$.*
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 some $i \in S_I$.*
3. *If $i_1 \overset{A_I}{\dashrightarrow}_I i_2$, $j_1 \overset{A_J}{\dashrightarrow}_J j_2$ and $(A_I \cap A_J) = M$, then $(i_1, j_1) \overset{A_I \cup A_J}{\dashrightarrow} (i_2, j_2)$.*

4. $i_1 \xrightarrow{\gamma}_I i_2$*, implies* $(i_1, j) \xrightarrow{\gamma} (i_2, j)$*, for some* $j \in S_J$*,*
5. $j_1 \xrightarrow{\gamma}_J j_2$*, implies* $(i, j_1) \xrightarrow{\gamma} (i, j)$*, for some* $i \in S_I$*,*

Rules $1.-3.$ apply to interactive transitions. The first two are for *independent* evolution of each connector (the other remaining in the same state). This independence is only allowed for transitions which do not interfere with the mixed nodes. This condition appears in a similar form in the definition of product of `Reo` automata, a model for non-stochastic `Reo`. Rule 3. defines joint evolution: if the nodes to be connected are ready to fire then they fire in both connectors. Rules $4.-5.$ are for Markovian transitions: evolution always happens interleaved. Figure 4 partially depicts the parallel composition of a lossy and a $\mathsf{fifo_e}$ channel with respect to a set $M = \{b\}$. We use a bar to separate the elements of the pair. Due to space limitations and readability issues, we do not present the full composition, which computes an $\mathtt{IMC_{Reo}}$ with more than 100 states.

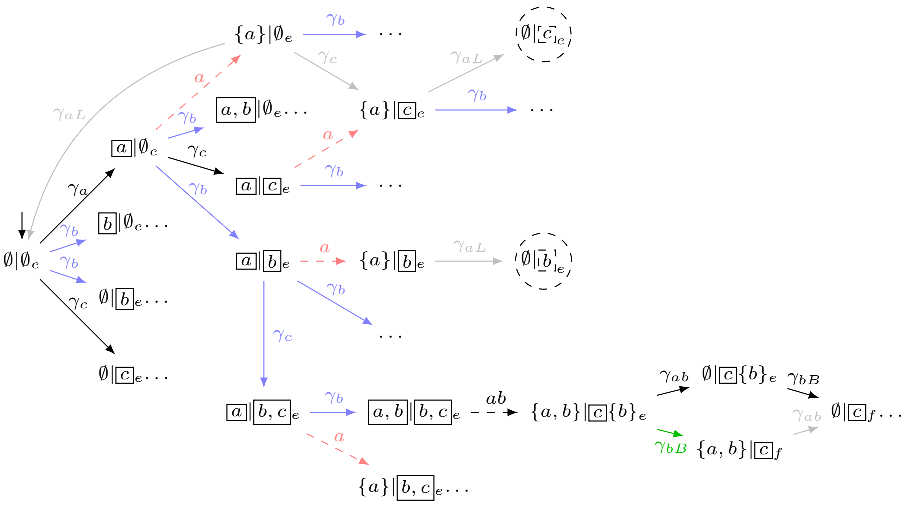

Fig. 4. Parallel Composition of a lossy and $\mathsf{fifo_e}$

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 a *self-contained pumping station* [2]. Failing to take this into account generates unwanted behaviour, making the semantics unsound. For example, the composition of a lossy and a $\mathsf{fifo_e}$, in Figure 4 allows for the data token arriving to port a to be lost, even if the buffer is empty. This corresponds to transition $\boxed{a}|\boxed{b}_e \overset{a}{\dashrightarrow} \{a\}|\boxed{b}_e$. But such a transition violates the mixed node assumption in the `Reo` *rationale* in the sense that, port a fires when port b is explicitly blocked.

Mixed nodes do not receive requests...
SELF-CONTAINED PUMPING STATION

The following definition captures this notion of *active blocking*. For notational convenience consider that, given a node $i = (R, T, Q)$ and a set of ports M $i\restriction_M$

refers to the node where all ports in M are considered hidden, *i.e.*, $i{\upharpoonright_M} = (R \setminus M, T, Q)$. For a composed node (i, j), the obvious pairwise extension $(i, j){\upharpoonright_M} = (i{\upharpoonright_M}, j{\upharpoonright_M})$ is used.

Definition 4 (Active blocking).
Given an $\texttt{IMC}_{\texttt{Reo}}$ $I = (S_1 \times S_2, Act, \dashrightarrow, \longrightarrow, i)$, *a node* (i, j) actively blocks *a set of nodes* M *if there exists a transition* $(i, j) \overset{X}{\dashrightarrow} (\_, \_)$ *with* $X \cap M = \emptyset$ *and*

- $R_i \cap M = \emptyset$ *and* $R_j \cap M \neq \emptyset$, *where* R_i *and* R_j *are the requests in* i *and* j, *respectively.*
  or
- *there exists* (i', j') *such that* $(i', j'){\upharpoonright_M} = (i, j){\upharpoonright_M}$ *and* $(i', j') \overset{Y}{\dashrightarrow} (\_, \_)$ *with* $X \cap Y = X$ *and typically* $Y \cap M \neq \emptyset$.

The first condition in the definition correspond to the active blocking explained above: for a port to fire, a mixed node in j is explicitly kept without firing. The second condition is another form of active blocking to which we call *forced nondeterminism*: there are two transitions in the chain that correspond to the same state modulo the presence of a request in the mixed node, but in one of them mixed nodes are explicitly blocked from firing, which again violates the *self-contained pumping station* assumption about mixed node behavior in Reo. As an example, in Figure 4, state $\boxed{a}|\boxed{c}$ actively blocks M, because $\boxed{a}|\boxed{c} \overset{a}{\dashrightarrow} \{a\}|\boxed{c}$ and there exists the state $\boxed{a,b}|\boxed{b,c}$ such that $\boxed{a,b}|\boxed{b,c}{\upharpoonright_M} = \boxed{a}|\boxed{c}{\upharpoonright_M}$, whose transition $\boxed{a,b}|\boxed{b,c} \overset{ab}{\dashrightarrow} \{a,b\}|\boxed{c}\{b\}$ holds $\{a, b\} \cap M \neq \emptyset$ and $\{a\} \cap \{a, b\} = \{a\}$ (the action of the blocking state).

We are now ready to introduce a synchronisation operation, which removes unwanted transitions from the chain and then prove, in Theorem 1, that it still preserves compositionality. Again, it should be remarked this operation is quite similar to the analogous one defined for `Reo` automata [13,14], which also hides mixed nodes.

Definition 5 (Synchronisation). *Given an* $\texttt{IMC}_{\texttt{Reo}}$ $I = (S_1 \times S_2, Act, \dashrightarrow, \longrightarrow, i)$, *we define the synchronisation of the chain with respect to a set of mixed nodes* M *by*

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

where

- $S_M = \{(i, j){\upharpoonright_M} |\ (i, j) \in S_1 \times S_2\}$
- *If* $i \overset{X}{\dashrightarrow} i'$, *and* i *does not actively block* M, *then* $i{\upharpoonright_M} \overset{X \setminus M}{\dashrightarrow}_M i'{\upharpoonright_M}$.
- *If* $i \xrightarrow{\gamma} i'$ *and* $R_{i'} \cap M = \emptyset$, *then* $i{\upharpoonright_M} \xrightarrow{\gamma} i'{\upharpoonright_M}$. *Here,* $R_{i'}$ *are the requests in* i'.

Finally, composition on mixed nodes M is defined as $\partial_M(I_1 \mathbin{||_M} I_2)$, where I_1, I_2 are two $\texttt{IMC}_{\texttt{Reo}}$.

In Figure 4, we display in red transitions that are deleted because the state is actively blocking the mixed node. We display in blue the transitions deleted by the second condition of the synchronisation, that is Markovian transitions to states with requests in the mixed node.

4.3 Compositionality

A compositionality result, stating that no matter in which order connectors are plugged their behavior is the same, can now be proved. Note that behaviour equivalence is the usual `IMC` bisimilarity, as defined in [23].

Theorem 1 (Compositionality). *Let* I_1 *and* I *be* $\texttt{IMC}_{\texttt{Reo}}$*, where* Act_1 *is the alphabet of* I_1*. The following holds:*

1. $\partial_M(I_1 \;||_{M_1} I) \sim I_1 \;||_{M_1} \partial_M I$*, if* $Act_1 \cap M = \emptyset$*.*
2. $\partial_{M_2}(\partial_{M_1} I) = \partial_{M_1}(\partial_{M_2} I) = \partial_{M_1 \cup M_2} I$*.*

Proof. For 1., note that a transition will be deleted from $I_1 \;||_{M_1} I$ if a node blocks behavior. However, because $Act_1 \cap M = \emptyset$, the deleted transition will also correspond to a node that blocks behavior in I. Hence, in the parallel composition, every transition with blocking source state (i, j) will be deleted if and only if the transition is also not present in $\partial_M I$. Which means such transition will also not be present in $I_1 \;||_{M_1} \partial_M I$. For Markovian transitions, the only kept are those which do not land in states with requests in M, which will be the same in both chains considered.

For 2., note that all interactive transitions in $\partial_{M_2}(\partial_{M_1} I)$ and $\partial_{M_1}(\partial_{M_2} I)$ are such that the source node does not actively block $M1$ and M_2. In other words $R_i \cap M_1 = \emptyset$, $R_i \cap M_2 = \emptyset$, $R_j \cap M_1 \neq \emptyset$ and $R_j \cap M_2 \neq \emptyset$. This is equivalent to $R_i \cap (M_1 \cup M_2) = \emptyset$ and $R_j \cap (M_1 \cup M_2) \neq \emptyset$. Hence, this corresponds to transitions whose source node does not actively block $M_1 \cup M_2$, which are all interactive transitions in $\partial_{M_1 \cup M_2} I$. For the Markovian transitions, the equality is a simple consequence of $(i\restriction_{M_1})\restriction_{M_2} = i\restriction_{M_1 \cup M_2}$.

4.4 Sequencing and unintended request arrivals

In general, when `Reo` channels are set in parallel within a connector, they evolve independently. However, when connected on their ends, data flows from channel to channel in sequence and there is a clear intended *direction* of flow. Similarly to many models, we have so far simplified the analysis and we have not explicitly modeled the difference between input and output ports, which then set the data flow direction. This generates some imprecisions. In Figure 4, node ${\scriptstyle \{a,b\}|\boxed{c}\{b\}_e}$ evolves interleaved via γ_{ab} or γ_{bB} to the same state. However, this allows for the buffer to become full before data is transmitted through the `lossy` channel, which is not intended. Moreover, when a channel is transmitting data from a port to another, arrival of requests might not be desired (this is of course subject of discussion, arriving requests could also be stored), because ports are busy. Note that this problem is only occurring in the Markov transitions.

To solve this, the following can be done. Given an $\texttt{IMC}_{\texttt{Reo}}$ $I = (S_1 \times S_2, Act, \dashrightarrow, \longrightarrow, i)$, we explicitly model the direction of flow by considering that Act is equipped with a partial order $<$. That is, given two ports $a, b \in Act$ if $a < b$ then data flows from a to b or, in other words, first in a and then in b. Given this, we can define when an $\texttt{IMC}_{\texttt{Reo}}$ respects sequencing.

Definition 6 (Sequencing). *Given an* $\mathtt{IMC}_{\mathtt{Reo}}$ $I = (S, Act, \dashrightarrow, \longrightarrow, i)$*, where the set of labels is equipped with a partial order* $<$*, we say that* I *respects sequencing if for every transition* $i \xrightarrow{\gamma} f$*, the set of nodes that finished transmission in this transition, that is* $T_i \setminus T_f$*, does not contain any element greater than any element in the set of nodes that still needs to transmit, that is, the elements of* T_f*.*

Similarly, we can also define when an $\mathtt{IMC}_{\mathtt{Reo}}$ respects no arrival requests on busy nodes.

Definition 7 (No arrivals when busy).
Given an $\mathtt{IMC}_{\mathtt{Reo}}$ $I = (S, Act, \dashrightarrow, \longrightarrow, i)$*, we say that* I *does not allow requests on busy nodes if for every transition* $i \xrightarrow{\gamma} f$*, the set of nodes that have a request after this transition is taken, that is* $R_{i'}$*, does not overlap with the set of active nodes.*

The set of active nodes for each state is easily obtained from the set of nodes effectively active (T_i) and their relations from the assumed partial order by taking advantage of its transitivity property. This set of active nodes represent, in fact, the nodes that transmit in each synchronous (atomic) data flow in `Reo`.

In order to incorporate these into the framework, one can define at a final stage of composition a *clean-up* operation that deletes all the transitions that do not comply with these properties (in case these properties are wanted). Compositionality will not be affected, because all the transitions that would be deleted in smaller components will also be deleted in larger components including these.

In Figure 4, we represent in green transitions that would be deleted by these properties. In grey are transitions deleted because their source states are unreachable.

4.5 Composition examples

To now illustrate the full process of composition of $\mathtt{IMC}_{\mathtt{Reo}}$, we present three different examples, each one with different subtleties, which are worth pointing out. Figure 5 results from composing a lossy and a sync channel.

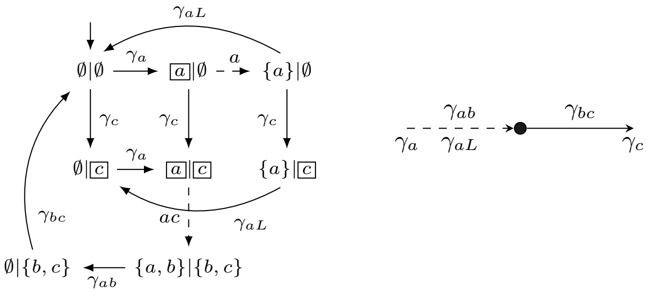

Fig. 5. The $\mathtt{IMC}_{\mathtt{Reo}}$ for the composition of lossy and sync channels.

Note that the result is an $\mathtt{IMC}_{\mathtt{Reo}}$ corresponding to a lossy channel with ports a and c and processing delay resulting from the relevant operation that compose rates γ_{ab} and γ_{bc}. This illustrates that the sync channel behaves as the identity of $\texttt{IMC}_{\textsf{Reo}}$ composition, as, in fact, it is expected in Reo.

Figure 6 presents the composition of a lossy and a $\textsf{fifo}_{\textsf{e}}$ channel. This example shows that data is not lost when the buffer is empty, unlike what happens, for instance, with constraint automata as stressed in [14].

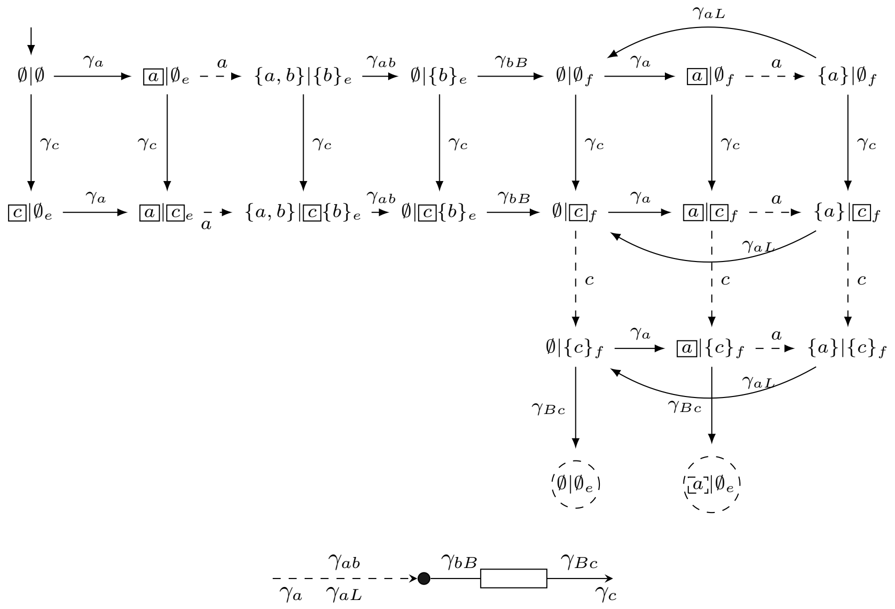

Fig. 6. The $\texttt{IMC}_{\textsf{Reo}}$ for the composition of lossy and $\textsf{fifo}_{\textsf{e}}$ channels.

Finally, Figure 7 partially depicts the $\texttt{IMC}_{\textsf{Reo}}$ for the composition of two $\textsf{fifo}_{\textsf{e}}$. We show in this example that when the first buffer is full and the second is empty, then data may flow instantaneously to the second buffer, freeing the first one. The τ-transitions, which appear by hiding mixed node b, explicitly model this intended behaviour. Consequently, the *maximal progression assumption* would simplify this chain by deleting Markovian transitions leaving unstable states.

5 Conclusions

In this paper, we proposed Interactive Markov chains (IMC) as a semantic model for stochastic Reo. This has several advantages to existing models. It does not use an intermediate automata model, which avoids extra translation steps. Furthermore, it is a compositional model which is important for behavioural but also for efficiency/implementation purposes, since it enables local changes to the circuit without having to recalculate the whole model. Last, but certainly not least, IMC are a well-studied formalism with many tools and results surrounding their theory. Casting stochastic Reo within IMC opens the door to all of these.

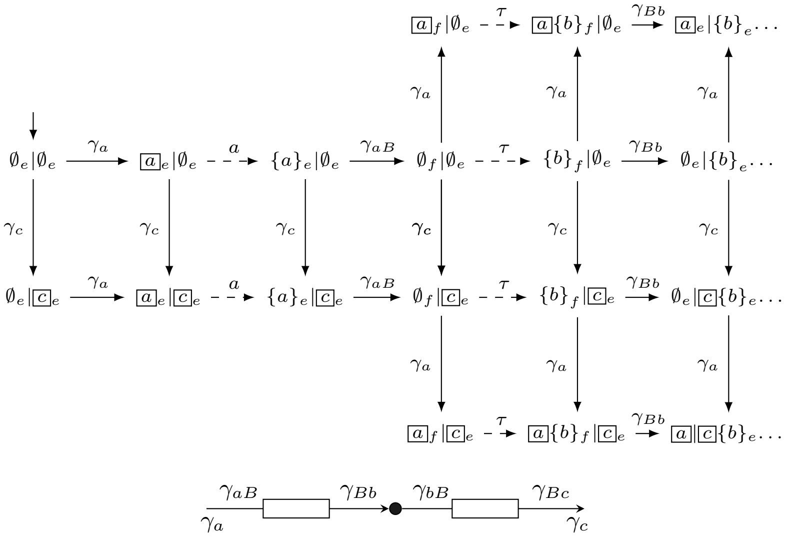

Fig. 7. The partial $\mathtt{IMC}_{\mathtt{Reo}}$ for the composition of two $\mathsf{fifo}_{\mathsf{e}}$ channels.

The simplification of mixed nodes as *self-pumping* stations that allow for data to be read and written with no processing delay was present since the invention of `Reo` and kept when the stochastic version was designed. However, this feature is not desired in practice. Such `I/O` operations take time and, therefore, may interfere with `QoS` values. In order to incorporate this in the model, and achieve an even more modular and faithful model, the following could be done. Each channel would have its boundary nodes modeled as independent stochastic processes with a processing delay rate. The *inside* of the channel would also be an independent process with a processing delay rate and a data transmission policy. Composition would then be achieved by taking all of these small components together. The tools developed in this paper, like the synchronisation operation, could be directly used for this purpose.

ECT [5] offers a plugin-based integrated environment to model and analyse `Reo` coordination. As a consequence of the work presented, tools to translate basic `Reo` channels into `IMC` and to perform their composition and synchronisation are in order. On the other hand, tools like CADP [19], PRISM [24,26] allow for both the qualitative/quantitative analysis and modelling of distributed stochastic processes. In particular, CADP is able to compose `IMC` and IMCA [20] is specifically designed for the analysis of `IMC`. A chain of such tools for efficient stochastic analysis of $\mathtt{IMC}_{\mathtt{Reo}}$ (the necessary output from ECT) is not only a desired feature but also effortlessly achieved[3].

Acknowledgments. We thank Farhad Arbab for several suggestions and comments.

---

[3] Implementation details and information about $\mathtt{IMC}_{\mathtt{Reo}}$ may be seen in `http://reo.project.cwi.nl/reo/wiki/ImcReo`

References

1. Farhad Arbab. Abstract behavior types: A foundation model for components and their composition. In Frank S. de Boer et al., eds, FMCO, LNCS 2852, pp. 33–70, Springer, 2003.
2. Farhad Arbab. Reo: a channel-based coordination model for component composition. *Mathematical. Structures in Comp. Sci.*, 14(3):329–366, June 2004.
3. Farhad Arbab. Composition by interaction. Leiden University, October 2005.
4. Farhad Arbab, Tom Chothia, Rob van der Mei, Sun Meng, YoungJoo Moon, and Chrétien Verhoef. From coordination to stochastic models of QoS. In John Field et al., eds, *Coordination'09*, LNCS 5521, pp. 268–287, Springer, 2009.
5. Farhad Arbab, Christian Krause, Ziyan Maraikar, Young-Joo Moon, and José Proença. Modeling, testing and executing reo connectors with the eclipse coordination tools. In *proceedings of FACS 2008*, September 2008.
6. Farhad Arbab and Farhad Mavaddat. Coordination through channel composition. In Farhad Arbab et al., eds, *Coordination'02*, LNCS 2315,Springer, 2002.
7. Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert Brayton. Model-checking continuous-time markov chains. *ACM Trans. Comput. Logic*, 1:162–170, July 2000.
8. J. C. M. Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2-3):131–146, May 2005.
9. Christel Baier, Boudewijn Haverkort, Holger Hermanns, and Joost P. 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 J. M. M. Rutten. Modeling component connectors in reo by constraint automata. *Sci. Comput. Program.*, 61(2):75–113, 2006.
11. Christel Baier and Verena Wolf. Stochastic reasoning about channel-based component connectors. In *Coordination'06*, pp. 1–15, Springer, 2006.
12. Marco Antonio Barbosa, Luís Soares Barbosa, and José Creissac Campos. Towards a coordination model for interactive systems. *Electronic Notes in Theoretical Computer Science*, 183:89–103, 2007.
13. Marcello Bonsangue, Dave Clarke, and Alexandra Silva. Automata for Context-Dependent connectors. In *Coordination'09*, LNCS 5521, pp. 184–203, Springer, 2009.
14. Marcello M. Bonsangue, Dave Clarke, and Alexandra Silva. A model of context-dependent component connectors. *Sci. Comput. Program.*, 77(6):685–706, 2012.
15. Dave Clarke, David Costa, and Farhad Arbab. Connector colouring I: Synchronisation and context dependency. *Science of Computer Programming*, 66(3):205–225, 2007.
16. David Costa. *Formal Models for Component Connectors.* PhD thesis, Vrije University, Amsterdam, October 2010.
17. José Luiz Fiadeiro and Antónia Lopes. Community on the move: Architectures for distribution and mobility. In Frank S. de Boer et al., eds, *FMCO*, LNCS 3188, pp. 177–196, Springer, 2003.
18. Cédric Fournet and Georges Gonthier. The Join calculus: A language for distributed mobile programming. In Gilles Barthe, Peter Dybjer, Luis Pinto, and João Saraiva, eds, *APPSEM*, LNCS 2395, pp. 268–332, Springer, 2000.
19. 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 (STTT)*, pp. 1–19, 2012.

20. Dennis Guck, Tingting Han, Joost-Pieter Katoen, and MartinR Neuhäußer. Quantitative timed analysis of interactive markov chains. In Alwyn Goodloe and Suzette Person, eds, *NASA Formal Methods*, LNCS 7226, pp. 8–23. Springer, 2012.
21. Juan Visente Guillen Scholten. *Mobile channels for exogenous coordination of distributed systems: semantics, implementation and composition.* PhD thesis, LIACS, Faculty of Mathematics and Natural Sciences, Leiden University, January 2007.
22. Holger Hermanns. *Interactive Markov chains: and the quest for quantified quality.* Springer, 2002.
23. Holger Hermanns and Joost P. Katoen. The how and why of interactive Markov chains. In *Proceedings FMCO'09*, pp. 311–337, Springer, 2010.
24. Andrew Hinton, Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM: A tool for automatic verification of probabilistic systems. In H. Hermanns and J. Palsberg, eds, *Proc. of TACAS'06*, LNCS 3920, pp. 441–444, Springer, 2006.
25. Mohammad Izadi, Marcello M. Bonsangue, and Dave Clarke. Modeling component connectors: Synchronisation and context-dependency. In Antonio Cerone and Stefan Gruner, eds, *SEFM*, pp. 303–312. IEEE Computer Society, 2008.
26. Marta Kwiatkowska, Gethin Norman, and David Parker. A framework for verification of software with time and probabilities. In K. Chatterjee and T. Henzinger, eds, *Proc. of FORMATS'10*, LNCS 6246, pp. 25–45, Springer, 2010.
27. R. Milner. *A Calculus of Communicating Systems.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.
28. Misra, Jayadev, Cook, and William. Computation orchestration: A basis for wide-area computing. *Software and Systems Modeling (SoSyM)*, 6(1):83–110, March 2007.
29. Young-Joo Moon, Alexandra Silva, Christian Krause, and Farhad Arbab. A compositional semantics for stochastic reo connectors. In *Proceedings FACS'10*, pp. 93–107, 2010.
30. 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*, 2012. To appear.
31. Oscar Nierstrasz. Piccola – a small compositional language (invited talk). In Paolo Ciancarini, Alessandro Fantechi, and Roberto Gorrieri, eds, *FMOODS*, volume 139 of *IFIP Conference Proceedings*. Kluwer, 1999.