Quantitative Analysis of Reo-based Service Coordination

Nuno Oliveira*
HASLab INESC TEC
Universidade do Minho
Braga, Portugal
nunooliveira@di.uminho.pt

Alexandra Silva
Centrum Wiskunde & Informatica
Radboud University Nijmegen
Amsterdam, The Netherlands
alexandra@cs.ru.nl

Luís S. Barbosa
HASLab INESC TEC
Universidade do Minho
Braga, Portugal
lsb@di.uminho.pt

ABSTRACT

Quality of Service analysis of composed software systems is an active research area, with the goal of evaluating and improving performance and resource allocation in service-oriented applications, namely, in the glue code –coordination layer– of such systems. Stochastic Reo offers constructs for service coordination and allows the specification of stochastic values for channels. But its state-of-the-art semantic models fail in several (important) ways. In this paper, we will see how Interactive Markov chains (IMC), proposed as a stochastic compositional model of concurrency, can be effectively used to serve as a compositional semantic model for Stochastic Reo. Treating IMC as a direct semantic model, gives rise to more faithful models and has obvious efficiency advantages. Moreover, tool support that exists for IMC is made available, without significant effort, to verify and reason about the coordination layer modelled as Reo connectors.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; D.2.8b [**Software Engineering**]: Performance measures

General Terms

Service Coordination Analysis

Keywords

Interactive Markov chains, Reo, Coordination, Analysis

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 [20, 13, 22], usually in a scripting language, whereas others offer a more visual approach, where 'channels' or 'connectors' are used to compose components into a system [8, 24, 1, 12].

Channel based-languages play a prominent role in the world of software composition. One of such languages is Reo [1],which offers a model of component and service coordination, wherein complex connectors are constructed by composing various types of primitive channels. Stochastic Reo [**?**] 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 [6, **?**, 9, 11, 10, **?**, **?**], 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 [7], stochastic intensional automata [2], and stochastic Reo automata [**?**, **?**]. 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 [**?**], 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 not to 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

*This author is supported by an Individual Doctoral Grant from FCT, with reference SFRH/BD/71475/2010.

SAC'14 March 24-28, 2014, Gyeongju, Korea.

shown to not be suitable for many Reo models.

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 [14] or IMCA [15] 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.

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 compositionality result. Section 5 presents tools to deal with the introduced model and shows its effectiveness with an example. We present concluding remarks and directions for future work in Section 6.

2. BACKGROUND

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

2.1 Reo and Stochastic Reo

Reo [?, ?, 1] 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 compositional 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.

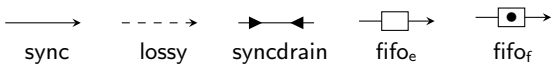

Figure 1: Primitive Reo channels.

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.

Stochastic Reo [2, ?] 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

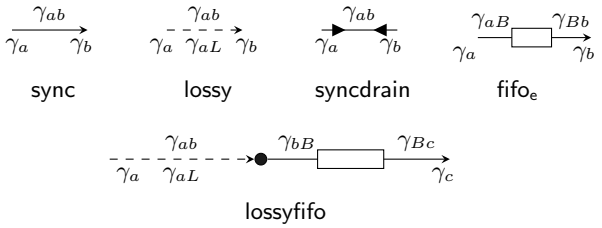

Figure 2: Primitive stochastic Reo channels.

Reo is still compositional. 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.

2.2 Interactive Markov Chains

Interactive Markov Chains (IMC) [16, 17] were proposed as a model for performance evaluation of distributed systems. The approach combines systems quantitative modelling, based on continuous-time Markov chains (CTMC) [5, 4], and process algebra [19, ?], to ensure compositionally.

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 activities. 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. 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 and $s \in S$ is the initial state of the chain.

Markovian transitions (s, γ, s') are denoted $s \xrightarrow{\gamma} s'$ and represent a transition from state s to state s' within t time units with a probability of $1 - e^{-\gamma \cdot t}$. 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 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 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 leaving the same state (known as *unstable state*), because the probability to execute such transition 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*. Note that this only concerns Markovian transitions; interactive transitions may as well execute immediately.

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 [10].

Before introducing our proposal for an IMC model for stochastic Reo, referred to as IMC_{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 $R, T \in \mathcal{P}(\mathcal{N})$ denote the sets of ports with, respectively, pending requests and data being transmitted. Naturally, the empty set, $\emptyset$, represents absence of requests and transmissions; and $Q \in \mathcal{Q}$ is an internal state identifier. This is in fact only used to distinguish control states in state-based connectors, such as the fifo where, for instance, $\mathcal{Q} = \{\mathsf{empty}, \mathsf{full}\}$.

In the context of IMC modelling of (stochastic) Reo, Markovian transitions will be labelled by $\gamma \in \mathbb{R}^+$, representing the delays according to rate γ. 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 IMC_{Reo} modelling a Reo channel, is an instance of a classical IMC, with a structured set of states and labels. Formally,

Definition 2. An IMC_{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; $Act \subseteq \mathcal{P}(\mathcal{N})$ 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 and $s \in S$ 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 $\{T\}_Q$; 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 are denoted as usual and Interactive transitions $(s, \{a_1, a_2, ...\}, s')$ by $s \overset{a_1\, a_2 \cdots}{\dashrightarrow} s'$. 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 IMC_{Reo} for the basic stochastic Reo channels. For instance, the IMC_{Reo} of a stochastic sync chan-

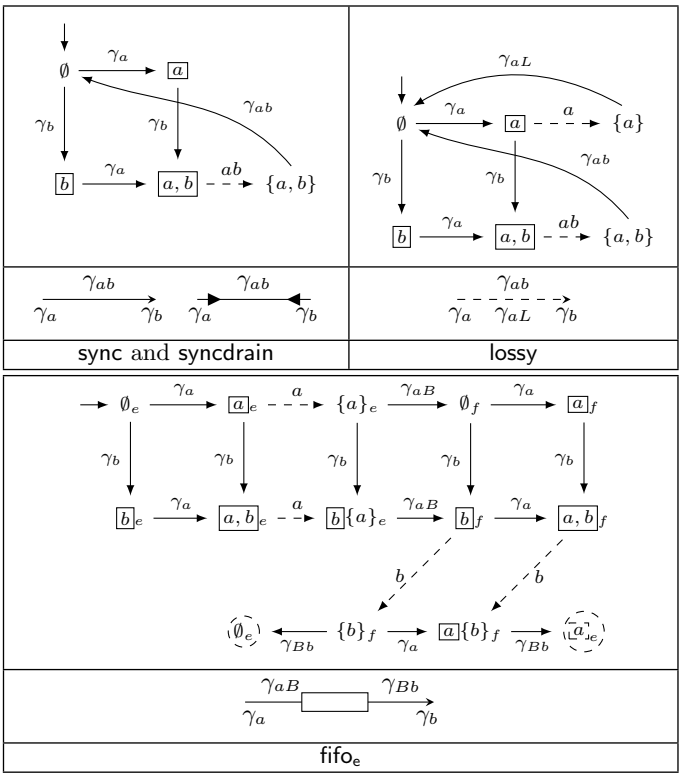

Figure 3: IMC for basic stochastic Reo channels

nel 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 [16], 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 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_{Reo} by explicitly dealing with sets of actions.

Definition 3. 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_{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 $j \in S_J$.
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 $(A_I \cap A_J) \subseteq M$, then $(i_1, j_1) \overset{A_I \cup A_J}{\dashrightarrow} (i_2, j_2)$.
4. $i_1 \overset{\gamma}{\longrightarrow}_I i_2$, implies $(i_1, j) \overset{\gamma}{\longrightarrow} (i_2, j)$, for $j \in S_J$,
5. $j_1 \overset{\gamma}{\longrightarrow}_J j_2$, implies $(i, j_1) \overset{\gamma}{\longrightarrow} (i, j)$, for $i \in S_I$,

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 depicts interesting parts (for further discussion) of the parallel composition between a lossy and a 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 IMC_{Reo} with 72 states and 182 transitions.

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* [1].

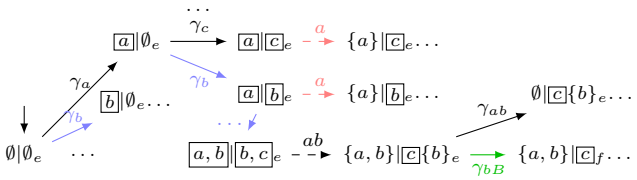

Figure 4: Parallel Composition of a lossy and fifo_e

Failing to take this into account generates unwanted behaviour, making the semantics unsound. For example, the composition of a lossy and a 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 $a|b_e \overset{a}{\dashrightarrow} \{a\}|b_e$. 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.

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.*, $i \restriction_M = (R \setminus M, T, Q)$. For a composed node (i, j), the obvious pairwise extension $(i, j)\restriction_M = (i\restriction_M, j\restriction_M)$ is used.

Definition 4. Given an IMC_{Reo} $(S_1 \times S_2, Act, \dashrightarrow, \longrightarrow, i)$, 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 $(i', j')\restriction_M = (i, j)\restriction_M$ and $(i', j') \overset{Y}{\dashrightarrow} (\_, \_)$ with $X \cap Y = X$ and $Y \cap M \neq \emptyset$.

The first condition in the definition corresponds 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 behaviour in Reo. As an example, in Figure 4, state $a|c$ actively blocks M, because $a|c \overset{a}{\dashrightarrow} \{a\}|c$ and there exists the state $a,b|b,c$ such that $a,b|b,c\restriction_M = a|c\restriction_M$, whose transition $a,b|b,c \overset{ab}{\dashrightarrow} \{a,b\}|c\{b\}$ 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 [9, 10], which also hides mixed nodes.

Definition 5. For an IMC_{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

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

Finally, composition on nodes M is defined as $\partial_M(I_1 \ ||_M \ I_2)$, 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 behaviour is the same, can now be proved. Note that behaviour equivalence is the usual IMC bisimilarity, as defined in [17].

THEOREM 1. *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 behaviour. However, because $Act_1 \cap M = \emptyset$, the deleted transition will also correspond to a node that blocks behaviour 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 Unintended Transitions

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 modelled the difference between input and output ports, which then set the data flow direction. This generates some imprecisions. In Figure 4, node $\{a,b\}|\boxed{c}\{b\}_e$ 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 markovian 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. Given an $\texttt{IMC}_{\texttt{Reo}}$ $I = (S, Act, \dashrightarrow, \longrightarrow, i)$, where the set of labels is equipped with a partial order $<$, we say that I respects sequencing if for every transition $i \longrightarrow^{\gamma} f$, 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 $\texttt{IMC}_{\texttt{Reo}}$ respects no arrival requests on busy nodes.

Definition 7. Given an $\texttt{IMC}_{\texttt{Reo}}$ $I = (S, Act, \dashrightarrow, \longrightarrow, i)$, we say that I does not allow requests on busy nodes if for every transition $i \longrightarrow^{\gamma} f$, 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.

4.5 Composition examples

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

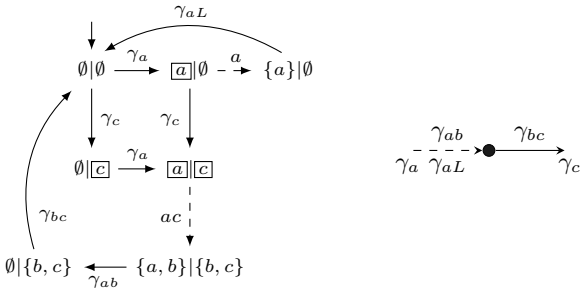

Figure 5: The $\texttt{IMC}_{\texttt{Reo}}$ for the composition of lossy and sync channels.

Note that the result is an $\texttt{IMC}_{\texttt{Reo}}$ 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 IMC_{Reo} composition, as, in fact, it is expected in Reo.

Figure 6 presents the composition of a lossy and a fifo_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 [10].

Due to space limitations, we just show a partial IMC_{Reo} for the composition of two fifo_e (the complete model has 39 states and 75 transitions). In this example we show that when the first buffer is full and the second is empty, 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 the chain by deleting Markovian transitions leaving unstable states.

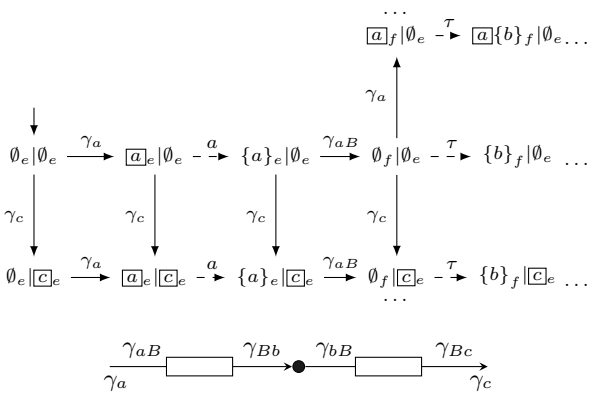

Figure 7: The partial IMC_{Reo} for the composition of two fifo_e channels.

5. TOOL SUPPORT

In order to be able to use the above model for a practical study, we developed a tool chain. The advantage of having IMC as the stochastic semantic model is that we build our tool set re-using developed tools. The tool chain is divided into two main parts: one for building the model, from the composition of several Reo channels, and another one for its analysis, linking to existing tools.

The first component (to which we refer to as IMCREOtools) takes as input a description of a Reo connector with stochastic information based on the CooPLa language [23]. It composes a (randomly selected) initial channel with the subsequent ones taking into account the results of Theorem 1 to avoid state space explosion and improve composition efficiency. This creates an IMC_{Reo}, which is then linearly converted into a standard IMC. The second component resorts to tools like IMCA, CADP or PRISM for the quantitative and qualitative analysis of the generated IMC.

For the purposes of this paper, because of space reasons, we will investigate the quantitative behaviour of two simple connectors, which contain a different number of buffers. The analysis will focus on the amount of data loss and its relation with the number of buffers. Though this is a toy example, it occurs in the context of more complex systems, e.g. the ASK systems [2]. Our two examples will be a lossyfifo with a single buffer (as presented in Figure 2), and another one with 4 buffers and similar stochastic properties ($\gamma_{ab} = 800$, $\gamma_{aL} = 450$, $\gamma_{bB} = 600$, $\gamma_{Bc} = 750$) measured for the same negligible time unit (t.u.). It is assumed a rate of 10^6 for the connections between buffers on the 4-buffer version of lossyfifo, which results in an almost instantaneous data transmission between buffers. Moreover, it is also assumed a constant consumption rate ($\gamma_c = 10$) and a variable production rate (γ_a ranging from 1 to 25).

To analyse the data losses in these two connectors and associated probabilities, we will use PRISM, which requires an extra translation step. PRISM does not support IMC and therefore we convert the obtained IMC into a CTMC, which can be done using the minimisation algorithms provided by CADP.

Hence, after modelling each connector in CooPLa, we used the *generator* tool from IMCREOtools to generate a CADP compatible model: the simple lossyfifo had 19 states and 33 transitions (as shown in Figure 6) and the 4-buffer version had 677 states and 2058 transitions. With the CADP *bcg_io* tool, we converted them into bcg models which were then minimised with the CADP *bcg_min* tool, creating two CTMC (the simpler one with 12 states and 20 transitions and the other one with 324 states and 972 transitions). Again, we converted the resultant bcg models into aut, with the *bcg_io* tool. To, finally, generate the PRISM input from the aut models, we used the *prismer* tool from IMCREOtools.

Within PRISM we analysed the behaviour of the two connectors by observing (*i*) the losses in the first 5 t.u., for a constant request arrival rate of 5 and (*ii*) the probability of losing data with respect to variations of the production rates (1 to 25) in the long run. The results are shown in Figure 8.

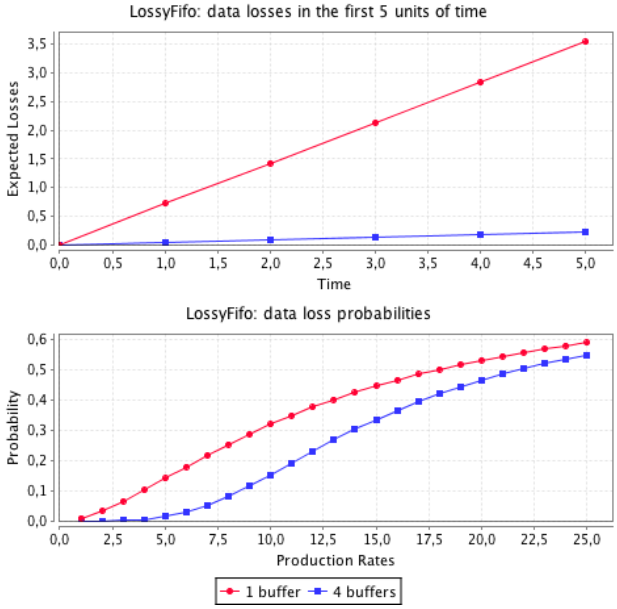

Figure 8: LossyFifo: data loss analysis

As expected, the amount of data lost after the first 5 t.u. (top graph) grows faster in a 1-buffer lossyfifo than in the 4-buffer version. From this analysis we may expect to lose 0.700 tokens per t.u. for the 1-buffer lossyfifo and 0.056

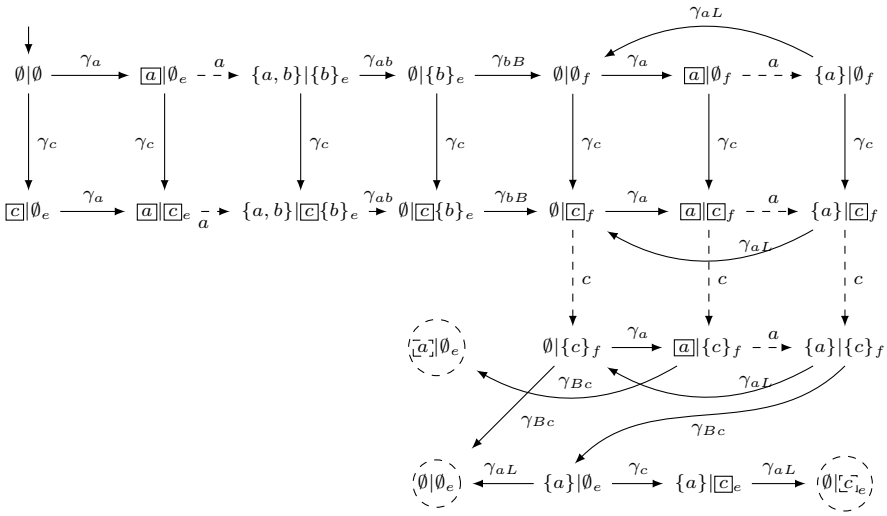

Figure 6: The $\texttt{IMC}_{\texttt{Reo}}$ for the composition of lossy and fifo_e channels.

tokens per t.u. for the 4-buffers version. This means an *average time to lose data* of 1.429 t.u. and 17.857 t.u. for the connectors with, respectively, 1-buffer and 4-buffers.

On the other hand, the long-run analysis (bottom graph) leads to interesting conclusions about the relations between the number of buffers and the production/consumption rates. To begin with, the 1-buffer lossyfifo converges faster to higher loss probabilities than its counterpart. For the 4-buffer version, it is possible to see that only when the arrival rate surpasses the number of buffers, the probability to lose data starts to increase more significantly. This means that the higher the number of buffers, the lower the probability to lose data. However, when the production rates tend to the infinity (and the consumption rate remains constant) the number of buffers becomes insignificant as the probability to lose data will tend to 1.

Albeit its simplicity and not very surprising analysis, this example illustrates our toolset and the value of the $\texttt{IMC}_{\texttt{Reo}}$ semantic model that we propose in this paper. The tool chain allows for a diversified functional and performance analysis, overcoming the problems of other models. An important point to note is that the tool chain scales up. This fact is proven by the existing case studies with PRISM and CADP showing that such tools can analyse chains with millions of states.

6. CONCLUSIONS

The paper proposed Interactive Markov Chains (IMC) as a semantic model for stochastic Reo. This has several advantages to existing models, from which we highlight three: it does not use an intermediate automata model, which avoids extra translation steps; it is a compositional model, which is important not only for behavioural but also for efficiency/implementation purposes; and last, but certainly not least, IMC are a well-studied formalism with many tools and results surrounding their theory. In particular measures typically obtained from the analysis of IMC become available for the study of stochastic Reo circuits. We developed IMCREOtools[1] to translate Reo connectors into $\texttt{IMC}_{\texttt{Reo}}$. The output of this tool is compatible with the most known tools to deal with IMC, e.g. CADP [14] and PRISM [18], which allow for both the qualitative/quantitative analysis and modelling of distributed stochastic processes; and IMCA [15] which is specifically designed for the analysis of IMC.

We see several directions for future work. 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 the following could be done. Each channel would have its boundary nodes modelled 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 notions developed in this paper, like the synchronisation operation, could be directly used for this purpose.

ECT [3] offers a plugin-based integrated environment to model and analyse Reo coordination. Providing a translation tool that directly interacts with ECT will enable us to analyse larger connectors which have been modelled in the past within ECT. It will also provide the Reo toolset with a range of quantitative tools.

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.

7. REFERENCES

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

[1] F. Arbab. Reo: a channel-based coordination model for component composition. *Mathematical. Structures in Comp. Sci.*, 14(3):329–366, June 2004.

[2] F. Arbab, T. Chothia, R. van der Mei, S. Meng, Y. Moon, and C. Verhoef. From coordination to stochastic models of QoS. In J. Field and V. Vasconcelos, editors, *Coordination Models and Languages*, volume 5521 of *LNCS*, chapter 14, pages 268–287. Springer, Berlin, Heidelberg, 2009.

[3] F. Arbab, C. Krause, Z. Maraikar, Y. Moon, and J. Proença. Modeling, testing and executing reo connectors with the eclipse coordination tools. In *proceedings of the International Workshop on Formal Aspects of Component Software (FACS 2008)*, Salamanca, Spain, September 2008.

[4] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model-checking continuous-time markov chains. *ACM Trans. Comput. Logic*, 1:162–170, July 2000.

[5] C. Baier, B. Haverkort, H. Hermanns, and J. P. Katoen. Model-Checking algorithms for Continuous-Time markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.

[6] C. Baier, M. Sirjani, F. Arbab, and J. J. M. M. Rutten. Modeling component connectors in reo by constraint automata. *Sci. Comput. Program.*, 61(2):75–113, 2006.

[7] C. Baier and V. Wolf. Stochastic reasoning about channel-based component connectors. In *Proceedings of the 8th international conference on Coordination Models and Languages*, COORDINATION'06, pages 1–15, Berlin, Heidelberg, 2006. Springer.

[8] M. A. Barbosa, L. S. Barbosa, and J. C. Campos. Towards a coordination model for interactive systems. *Electron. Notes Theor. Comput. Sci.*, 183:89–103, July 2007.

[9] M. M. Bonsangue, D. Clarke, and A. Silva. Automata for Context-Dependent connectors. In *Proceedings of the 11th International Conference on Coordination Models and Languages*, COORDINATION '09, pages 184–203, Berlin, Heidelberg, 2009. Springer.

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

[11] D. Costa. *Formal Models for Component Connectors*. PhD thesis, Vrije University, Amsterdam, October 2010.

[12] J. L. Fiadeiro and A. Lopes. CommUnity on the move: Architectures for distribution and mobility. In F. S. Boer, M. M. Bonsangue, S. Graf, and W. Roever, editors, *Formal Methods for Components and Objects*, volume 3188 of *LNCS*, pages 177–196. Springer, 2004.

[13] C. Fournet and G. Gonthier. The join calculus: A language for distributed mobile programming. In G. Barthe, P. Dybjer, L. Pinto, and J. Saraiva, editors, *Applied Semantics*, volume 2395 of *LNCS*, pages 268–332. Springer, 2002.

[14] H. Garavel, F. Lang, R. Mateescu, and W. Serwe. CADP 2011: a toolbox for the construction and analysis of distributed processes. *International Journal on Software Tools for Technology Transfer (STTT)*, pages 1–19, 2012.

[15] D. Guck, T. Han, J. Katoen, and M. R. Neuhäußer. Quantitative timed analysis of interactive markov chains. In A. E. Goodloe and S. Person, editors, *NASA Formal Methods*, volume 7226 of *LNCS*, pages 8–23. Springer, 2012.

[16] H. Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*, volume 2428 of *LNCS*. Springer, 2002.

[17] H. Hermanns and J. P. Katoen. The how and why of interactive markov chains. In *Proceedings of the 8th international conference on Formal methods for components and objects*, FMCO'09, pages 311–337, Berlin, Heidelberg, 2010. Springer.

[18] M. Kwiatkowska, G. Norman, and D. Parker. A framework for verification of software with time and probabilities. In K. Chatterjee and T. Henzinger, editors, *Proc. 8th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'10)*, volume 6246 of *LNCS*, pages 25–45. Springer, 2010.

[19] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer, 1980.

[20] J. Misra and W. R. Cook. Computation orchestration: A basis for wide-area computing. *Software and Systems Modeling (SoSyM)*, 6(1):83–110, March 2007.

[21] Y. Moon, A. Silva, C. Krause, and F. Arbab. A compositional model to reason about end-to-end QoS in stochastic reo connectors. *Science of Computer Programming*, December 2011.

[22] O. Nierstrasz. Piccola - a small compositional language (invited talk). In *Proceedings of the IFIP TC6/WG6.1 Third International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS)*, pages 457–, Deventer, The Netherlands, The Netherlands, 1999. Kluwer, B.V.

[23] N Oliveira and L. S. Barbosa. Reconfiguration mechanisms for service coordination. In MauriceH Beek and Niels Lohmann, editors, *Web Services and Formal Methods*, volume 7843 of *LNCS*, pages 134–149. Springer, 2013.

[24] J. V. G. 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.