An Enhanced Model for Stochastic Coordination

Nuno Oliveira
HASLab - INESC TEC
Universidade do Minho, Braga, Portugal
nuno43549@gmail.com

Luis Soares Barbosa
HASLab - INESC TEC
Universidade do Minho, Braga, Portugal [*]
lsb@di.uminho.pt

Applications developed over the cloud coordinate several, often anonymous, computational resources, distributed over different execution nodes, within flexible architectures. Coordination models able to represent quantitative data provide a powerful basis for their analysis and validation. This paper extends $\mathsf{IMC}_{\mathsf{Reo}}$, a semantic model for Stochastic Reo based on interactive Markov chains, to enhance its scalability, by regarding each channel and node, as well as interface components, as independent stochastic processes that may (or may not) synchronise with the rest of the coordination circuit.

1 Introduction

The increasing ubiquity and complexity of cloud applications and their management brings research on coordination languages and models [8] up front as a main tool for design and analysis. On the one hand, this opens an interesting opportunity for formal methods; on the other it clearly challenges their scalability.

This paper addresses such a challenge from a specific stand point: that of the Reo coordination model [1, 2] and its stochastic version [3, 10]. In a previous paper [13], the authors, in collaboration with Alexandra Silva, proposed a semantic model for Stochastic Reo based on interactive Markov chains [9]. The model, known as $\mathsf{IMC}_{\mathsf{Reo}}$, is compositional and has the advantage of bringing to the coordination community a panoply of tools developed for quantitative analysis of probabilistic transition systems. Due to a rapid state explosion, the (use of the) model, however, does not scale up to the point of being really useful for analysis of big coordination scenarios, as found in typical cloud applications.

The paper starts in Section 2 with a brief review of Reo, its stochastic version, and $\mathsf{IMC}_{\mathsf{Reo}}$. Due to space restrictions such introductions are necessarily very short; the interested reader is referred to the relevant literature [11, 13] for details. Sections 3 and 4 introduce an enhanced model which smoothly extends $\mathsf{IMC}_{\mathsf{Reo}}$, increasing its ability to deal with bigger and more complex coordination protocols in a stochastic setting. The proposed model, called $\mathscr{D}\mathsf{IMC}_{\mathsf{Reo}}$, from *distilled* $\mathsf{IMC}_{\mathsf{Reo}}$, relaxes the basic Reo assumption on mixed nodes as self-pumping stations [2], which allowsfor data to be read and written with no processing delay. In practice, namely for cloud based applications, this assumption is unrealistic: I/O operations take time and, therefore, may interfere with QoS values. Finally, section 5 concludes.

2 Background

Reo. Reo [1, 2] is a channel-based model for the exogenous coordination of components in the context of component-based software. A channel is a directed communication mean with exactly two ends: a

[*]Luis S. Barbosa is supported by grant SFRH/BSAB/113890/2015 from FCT, the Portuguese Foundation for Science and Tecnhology. This research is financed by the ERDF COMPETE 2020 Programme within project POCI-01-0145-FEDER-00696, and by National Funds through FCT as part of project UID/EEA/50014/2013.

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 asynchronous, exhibiting memory capabilities or the possibility of specifying an ordering policy for content delivery. Moreover, a channel may also be lossy when it delivers some values but loses others depending on a specified policy. Figure 2 recalls the basic channels used in Reo, represented, however, in their stochastic version. 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 input/output requests. The qualifiers e or f refer to the channel internal state (either *empty* or *full*). Finally, the drain channel accepts data synchronously at both ends and loses it.

Channels are composed to define more complex coordination structures referred to as connectors. Composition of channels is made on their ends, giving rise to nodes. 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 to as the connector's ports. Figure 1 presents three such connectors.

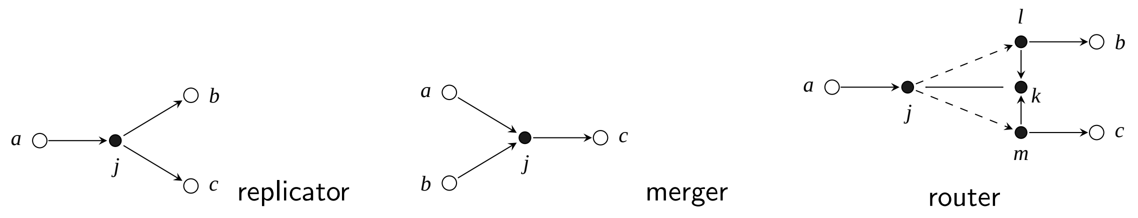


Figure 1: Reo connectors

As expected of any compositional model, Reo connectors behaviour arise from the behaviour of each constituent channel. However, as composition is made on channel ends, originating nodes, also these nodes contribute to the overall connector behaviour. The connectors of Figure 1 actually encode the simple form of three of these nodes. The replicator connector replicates data flowing from port a to ports b and c, in parallel, through mixed node j — the replicator node. This behaviour, which is synchronous, only holds when there are pending requests in all the connector ports. The merger connector merges data coming from ports a and b to port c, through mixed node j — the merger node. The merge of data 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 when there are pending requests at all the boundary ports, preventing one of the input ports from firing. The router connector, usually represented as $\otimes$, is a mutual exclusive router of data, taking data from input port a into either port b or port c, depending on the existence of pending requests at the output ports. When there are pending requests at the same time in both output ports, mixed node k — the router — non-deterministically choses (since it encodes a merge) which of the two ports will synchronously fire: either a and b or a and c.

Stochastic Reo. Stochastic Reo [3, 10] extends Reo by modelling coordination from a quantitative perspective. Non-negative real (stochastic) values are added both to channels and to their ends to represent, respectively, *processing delays* and IO *arrival rates*. The former models the time needed for the channel to process data from one point to another, where point refers to a channel end, a buffer or a point where data is lost or automatically produced. Each channel, depending on its type, may be annotated with more than one processing delays. Arrival rates model the time between consecutive arrivals of environment-issued IO operations to channel ends. Figure 2 shows the basic channels of stochastic Reo, represented as normal Reo channels, but annotated with stochastic values (rates and delays).

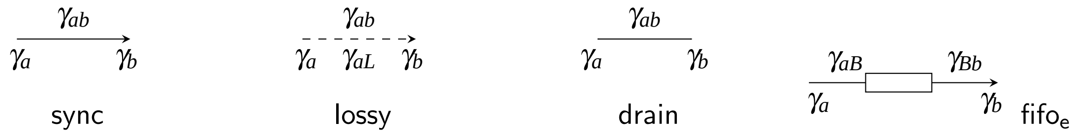


Figure 2: Primitive Stochastic Reo channels.

Stochastic Reo is still compositional. Processing delays of each individual channel in a composition scenario are not changed. The request arrival rates, however, are only preserved for the boundary nodes of the connector. As mixed nodes are internal (hidden from the exterior) the arrival request rates associated to the constituent channel ends are ignored, which means that these nodes are always ready to read/write data from/to the channels. This behaviour is known as the *self-contained pumping station*, firstly referred in [2].

$\mathsf{IMC_{Reo}}$. In a previous paper [13], the authors introduced a compositional semantic model for Stochastic Reo based on interactive Markov chains [9], a formalism combining continuous-time Markov chains[6, 4] with process algebra [5]. The model is state-based, states capturing the possible behaviour of a connector: data arrivals and data flowing through ports. Consider sets $\mathscr{N}$ and $\mathfrak{Q}$ of port names and internal state names, respectively. Each state in $\mathsf{IMC_{Reo}}$ is a triple (R,T,Q), where $R,T \in 2^{\mathscr{N}}$ denote sets of ports/nodes with, respectively, pending requests and data being transmitted; and $Q \in \mathfrak{Q}$ is an internal state identifier. The latter is used to distinguish between control states in state-based connectors. For example, in a fifo channel it may indicate whether the buffer is empty or full, by taking $\mathfrak{Q} = \{\mathsf{empty},\mathsf{full}\}$. Markovian transitions are labelled by $\gamma \in \mathbb{R}^+$. Distribution parameter γ encodes, in each case, the connector processing delays and the rates of data arrival at its ports. Interactive transitions, on the other hand, are labelled with a set F of ports which, on firing, allow data to flow through them. Such ports correspond to the set of actions observable at the relevant $\mathsf{IMC_{Reo}}$ state. In the sequel, this set is referred to as *actions*, for simplicity. The decision to take sets of actions (rather than a single action) to label interactive transitions was crucial to correctly capture (atomic) synchrony in the semantics of Reo. In fact, ports firing synchronously to enable data flow are the rule rather than the exception in Reo. Formally,

Definition 1. *An $\mathsf{IMC_{Reo}}$ model is a tuple $(S, Act, \dashrightarrow, \longrightarrow, s)$, where $S \subseteq Act \times Act \times \mathfrak{Q}$ is a nonempty set of states; $Act \subseteq 2^{\mathscr{N}}$ is a set of actions (the alphabet); $\dashrightarrow \subseteq S \times Act \times S$ 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 $s \overset{a_1 a_2 \ldots}{\dashrightarrow} s'$ 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 *mixed* states and are depicted 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 3 depicts the $\mathsf{IMC_{Reo}}$ models corresponding to the basic Stochastic Reo channels. To simplify the picture, transition overlapping is generally avoided by the graphical replication of states suitably annotated with a dashed circle.

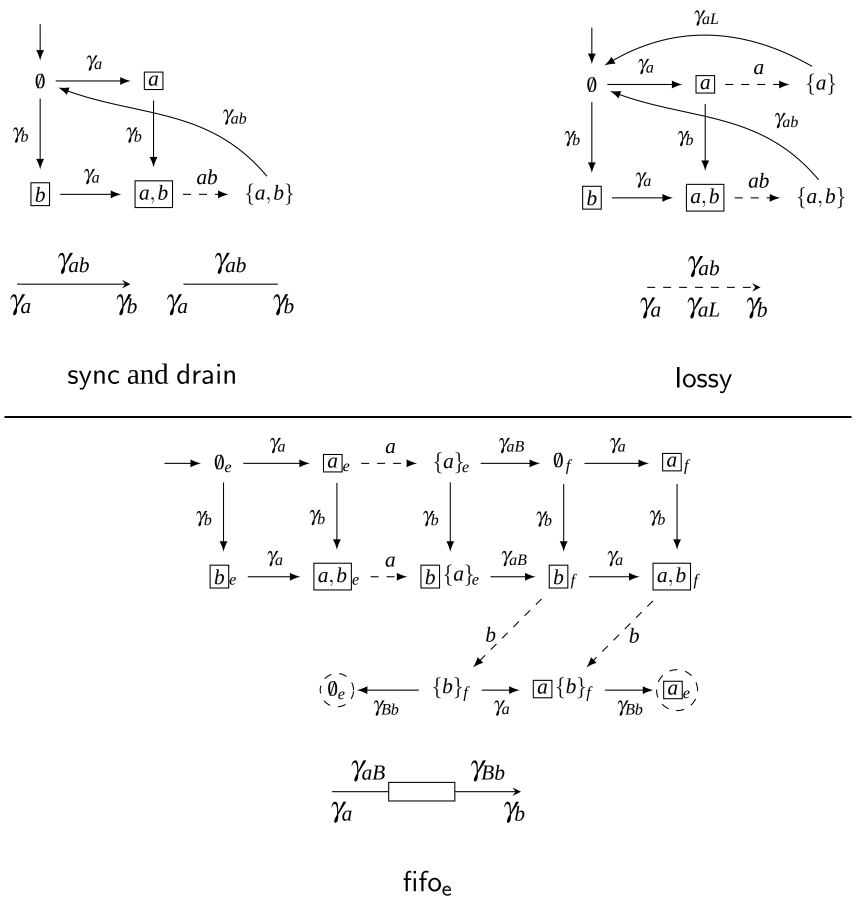


Figure 3: IMC for the basic stochastic Reo channels.

The $\mathsf{IMC_{Reo}}$ model of a stochastic sync channel is interpreted as follows: initially, no requests are pending neither in port *a* nor in port *b*. Requests 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 eventually fire. That is, actions *a* and *b* are activated simultaneously. At this moment, the channel starts transmitting data between *a* and *b* and evolves back to the initial state with a processing delay rate of γ_{ab}. For a stochastic lossy channel the interpretation is similar. However it exhibits two additional transitions to model the possibility of data being lost: at state $\boxed{a}$, port *a* may fire, because there is no pending request at port *b*. When such is the case, the channel evolves back to the initial state after a delay of discarding data. State $\boxed{a}$ captures the context-dependent behaviour characteristic of this channel. Finally, the fifo$_\mathsf{e}$ stochastic channel differs from the others 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 is the rule in $\mathsf{IMC_{Reo}}$

models.

The composition of two $\mathsf{IMC}_{\mathsf{Reo}}$ models I and J, with respect to a set of ports $M \subseteq \mathscr{N}$, is given by a product (which accounts for parallel evolution) and a synchronisation operation (which deals with interaction), and denoted by

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

The definitions of both operations are collected in the appendix; the reader is referred to [13] for examples and details. Note that this two-step composition approach is not a novelty in the definition of composition operations in Reo. Actually, it is very much in the same spirit of the one defined for Reo automata [7].

3 $\mathscr{D}\mathsf{IMC}_{\mathsf{Reo}}$: The new model

As mentioned in the Introduction, $\mathsf{IMC}_{\mathsf{Reo}}$ does not scale in a smooth way: composition generates a state space that remains considerably big even after minimisation via bisimulation. This limits its use for analysis of coordination, namely in the context of cloud-based systems involving an arbitrary number of actors.

Actually, as an exogenous coordination model, Reo disregards services or components when it comes to specifying a coordination schema. It only assumes that such computation *loci* are bound to the ports of the connector, which receive IO impulses whenever communication is requested. Consequently, Stochastic Reo inherits the same philosophy. But, does it? Not quite! In fact, Stochastic Reo circuits are not completely exogenous. They embody, in request arrival rates, information that is inherently associated to the induced stochastic behaviour of the interacting services coordinated by Stochastic Reo circuits. As expected, this hampers the reutilisation of Stochastic Reo models, and introduces unnatural simplifications to make it compositional.

As an alternative, we propose to consider the stochastic version of Reo as a two-phase component-based coordination model. The qualifier *two-phase* stresses the need for explicitly considering the model before and after deployment, known as the *design* and *deployment* phases, respectively; it is component-based because it is constructed from four specific components: the writer, the reader, the channel and the node, as graphically presented in Figure 4.

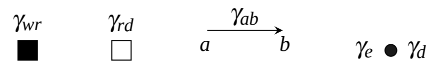


Figure 4: The essential components of Stochastic Reo.

The first two are synchronous stochastic abstractions of the real-world services that are to be bound to the ports of the connector. They are annotated with a delay rate (γ_{wr} and γ_{rd}, respectively), that models the time between consecutive IO requests issued by them. The channel component inherits the usual behaviour of Reo channels, as well as the processing delay rate of Stochastic Reo, which models the duration of point-to-point data transportation. Note that the request arrival rates are no more part of a channel model. The node is now taken as a synchronous component which behaves like the replicator, the merger or the router connector. Differently from the original version of Stochastic Reo, in this approach nodes are assumed to take time to enqueue and dequeue data. This behaviour is modelled by the delay rates γ_e and γ_d:

- Enqueueing data takes into account not only the time to process incoming data but also the time needed to select from which channel data will be read (if a merger);

- Dequeuing data takes into account the time to write data in the channels; it further comprises the time to generate copies of the data to write (in a replicator), and the time to decide to which channels it will write (in a router).

This captures a more realistic stochastic behaviour of nodes, as opposed to the usual *self-contained pumping station* behavioural assumption.

The design-phase models come from the composition of channel and node components. In turn, deployment-phase models are fixed for a given installation of composed services. The writer and the reader components are bound to the interface ports of the connector. This is, in fact, very close to the original Stochastic Reo model, adding to it, however, a more realistic separation of concerns. Figure 5 depicts a simple example of a lossyfifo connector in both the design- and the deployment-phase.

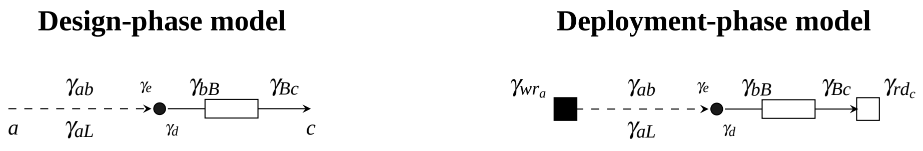


Figure 5: The two-pase, component-based model of a lossyfifo.

This component-based rephrasing of $\mathsf{IMC}_{\mathsf{Reo}}$ takes each channel, node, writer and reader as an independent stochastic process that may (or may not) synchronise with the other elements. The introduction of delays in nodes raises the need for two new sorts of states with specific semantics: the state where the node is enqueueing and the state where it is dequeueing data. A state in $\mathscr{D}\mathsf{IMC}_{\mathsf{Reo}}$ is fully characterised as (R,T,E,D,Q) with $E,D \in 2^{\mathscr{N}}$, where states of the form $(\emptyset,\emptyset,E,\emptyset,Q)$ are *enqueueing* states, *i.e.* in which the node is reading from the channel ends in set E; these states are represented as Ⓔ$_Q$. Likewise, states of the form $(\emptyset,\emptyset,\emptyset,D,Q)$ are *dequeueing* states, meaning that the node is writing to the channel ends in set D. These states are represented as Ⓓ$_Q$.

Apart from this modification on states, the basic formal model of $\mathsf{IMC}_{\mathsf{Reo}}$ remains unchanged, as well as the variants of bisimulation introduced in [13]. Let us, however, revisit the $\mathscr{D}\mathsf{IMC}_{\mathsf{Reo}}$ for each basic component.

Channels. The $\mathscr{D}\mathsf{IMC}_{\mathsf{Reo}}$ models for the basic Reo channels are depicted in Figure 6. They are obtained from their counterpart in $\mathsf{IMC}_{\mathsf{Reo}}$ models by disregarding the environment information. When compared to the corresponding $\mathsf{IMC}_{\mathsf{Reo}}$ representation, a significant reduction is visible in their state space.

Readers and writers. To obtain deployment-phase models it is necessary to compose design-phase models with the environment information, *i.e.* the reader and the writer components. Observationally, the latter would behave similarly: they issue IO requests by publishing the intention to write (respectively, read) data; then they block until synchronising with the connector ports. Thus, one single $\mathscr{D}\mathsf{IMC}_{\mathsf{Reo}}$ model is enough to capture such behaviour, as depicted in Figure 7.

A reader is bound to an output port while a writer is bound to an input port. This is how readers and writers are distinguished. The composition of these components with one channel will result in a $\mathscr{D}\mathsf{IMC}_{\mathsf{Reo}}$ model capturing the semantics of Stochastic Reo channels (and consequently, connectors).

Nodes. The basic Reo node ontology (replicator, merger and router) is extended to the six different configurations based over them, as shown in Figure 8.

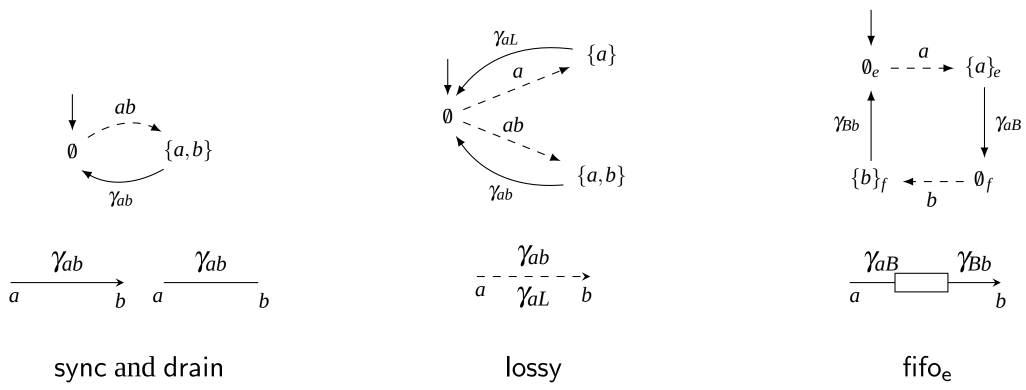


Figure 6: The $\mathscr{D}\mathsf{IMC}_{\mathsf{Reo}}$ models for basic Stochastic Reo channels.

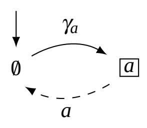


Figure 7: The $\mathscr{D}\mathsf{IMC}_{\mathsf{Reo}}$ for the reader and writer components

Note that node configurations (a) to (c) are special cases of (e): these nodes select one incoming channel to read data from, and then copy and write the data into all the outgoing channels. In turn, node configuration (d) is a special case of (f): it selects one incoming channel to read from, and then routes the data to one of the outgoing channels. Nodes (e) and (f) define, in fact, two families of nodes, referred henceforth as merger–replicator and merger–router, respectively. They are parametric on the number of incoming and outgoing channels and also on the delays for reading (enqueueing) and writing (dequeueing) data, whenever such delays are considered. Consistently, all $\mathscr{D}\mathsf{IMC}_{\mathsf{Reo}}$ nodes are generated from these two families, taking into account their parameters as follows:

$$\mathsf{merger\text{–}replicator}, \mathsf{merger\text{–}router} : 2^{\mathscr{N}} \times 2^{\mathscr{N}} \times \mathbb{R}^+ \times \mathbb{R}^+$$

where the first parameter is a set of output channel ends (the node inputs); the second is a set of input channel ends (its outputs); the third models the time to select and read from one channel end, and finally, the fourth parameter models the time to copy, route and write data into one channel end.

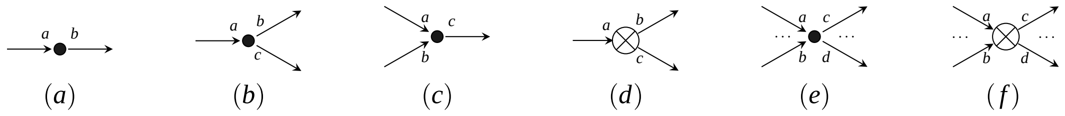
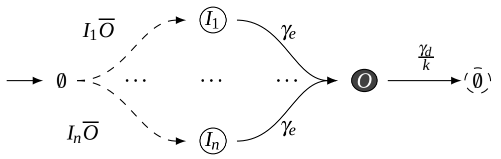
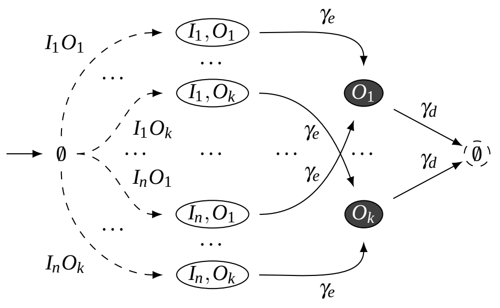


Figure 8: (a) simple; (b) replicator; (c) merger; (d) router; (e) merger-replicator: (f) merger-router.

Figure 9 depicts the parametric $\mathscr{D}\mathsf{IMC}_{\mathsf{Reo}}$ models for both the merger–replicator and the merger–router families of nodes. Notation I_i represents the i^{th} element in set I and $\overline{O}$ represents the concatenation of all elements in set O. Moreover, it is assumed that the cardinality of sets I and O are, respectively, n and k.





merger–replicator (I,O,γ_e,γ_d) merger–router (I,O,γ_e,γ_d)

Figure 9: $\mathscr{D}$IMC$_{\text{Reo}}$ models for merger–replicator and merger–router nodes.

The merger–replicator node blocks until synchronising with one of the input channel ends and all the output channel ends. On synchronisation, it starts enqueueing data from the input channel end (delayed for some exponentially distributed time modelled by γ_e). Then, it dequeues data to all the output channel ends and returns to the initial blocked state. The delay time of a single dequeue operation is exponentially distributed with rate γ_d; since it performs k such operations, then the average delaying time is exponentially distributed with rate $\frac{\gamma_d}{k}$. The merger–router, in turn, blocks until synchronising with one of the input and one of the output channels ends. On synchronisation, it goes to an enqueueing state and remains there for an exponentially distributed time modelled by rate γ_e. Then, it dequeues data to the selected output channel end at a rate γ_d, returning to the initial blocked state.

By disregarding enqueueing and dequeueing delays, these families of nodes are simplified into a single $\mathscr{D}$IMC$_{\text{Reo}}$ model with transition space size of n and $n.k$ for merger–replicator and merger–router, respectively, corresponding only to the interactive transitions.

4 Composition in $\mathscr{D}$IMC$_{\text{Reo}}$

Composition in $\mathscr{D}$IMC$_{\text{Reo}}$ extends that of IMC$_{\text{Reo}}$, adding to the parallel and synchronization steps (see Appendix), a phase for *cleaning* superfluous transitions which takes into account the need for enqueueing/dequeueing data in a specific order. Concretely, (i) data is always enqueued into the node only after being transmitted to that node; (ii) data is always transmitted to any further node only after being dequeued from the current one and (iii) data is always enqueued before being dequeued (from the same node). Actually, $\mathscr{D}$IMC$_{\text{Reo}}$ requires that enqueueing and dequeueing transitions appear immediately one after the other, except in cases where other operations may occur in parallel; when such is the case, transitions will appear interleaved. Formally, the cleaning operation is defined as follows:

Definition 2 ($\mathscr{D}$IMC$_{\text{Reo}}$ clean up)**.** *Let $M \subseteq \mathscr{N}$ and $I = (S, Act, \dashrightarrow, \longrightarrow, s)$ be a $\mathscr{D}$IMC$_{\text{Reo}}$. Assume also a relation $<$ on $\mathscr{N}$ such that $a < b$ when data flows from a to b, with $a,b \in \mathscr{N}$, which is lifted to sets as expected: $A < B$ iff $\exists_{a \in A} \cdot \forall_{b \in B} \cdot a < b$.*

The cleaning of I with respect to M, denoted $\mathscr{C}_M I$, corresponds to restricting $\partial_M I$ so that all its Markovian transitions $i \xrightarrow{\gamma} f$ respect:

(i) $R_f \cap AN(i) = \emptyset$, *where* $AN(i) = T_i \cup \{ j \in \mathscr{N} \mid \exists_{k \in T_i} . j < k \vee k < j \}$;

(ii) $$\begin{cases} T_i = T_f & \text{if} \quad E_i < T_i \quad \text{or} \quad T_i \cap D_i \neq \emptyset \\ T_i \setminus T_f < T_f & \text{otherwise} \end{cases}$$

and all its interactive transitions $j \overset{X}{\dashrightarrow} k$ *respect:*

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

The following example shows the (design-phase) composition of a lossy channel with a sync channel, considering that data enqueueing and dequeueing in the mixed node is delayed with rates γ_{enq} and γ_{deq}, respectively. Figure 10 depicts the composition of the two channels and the synchronising node. The greyed-out transitions are eliminated by cleaning, as they fail to respect sequencing.

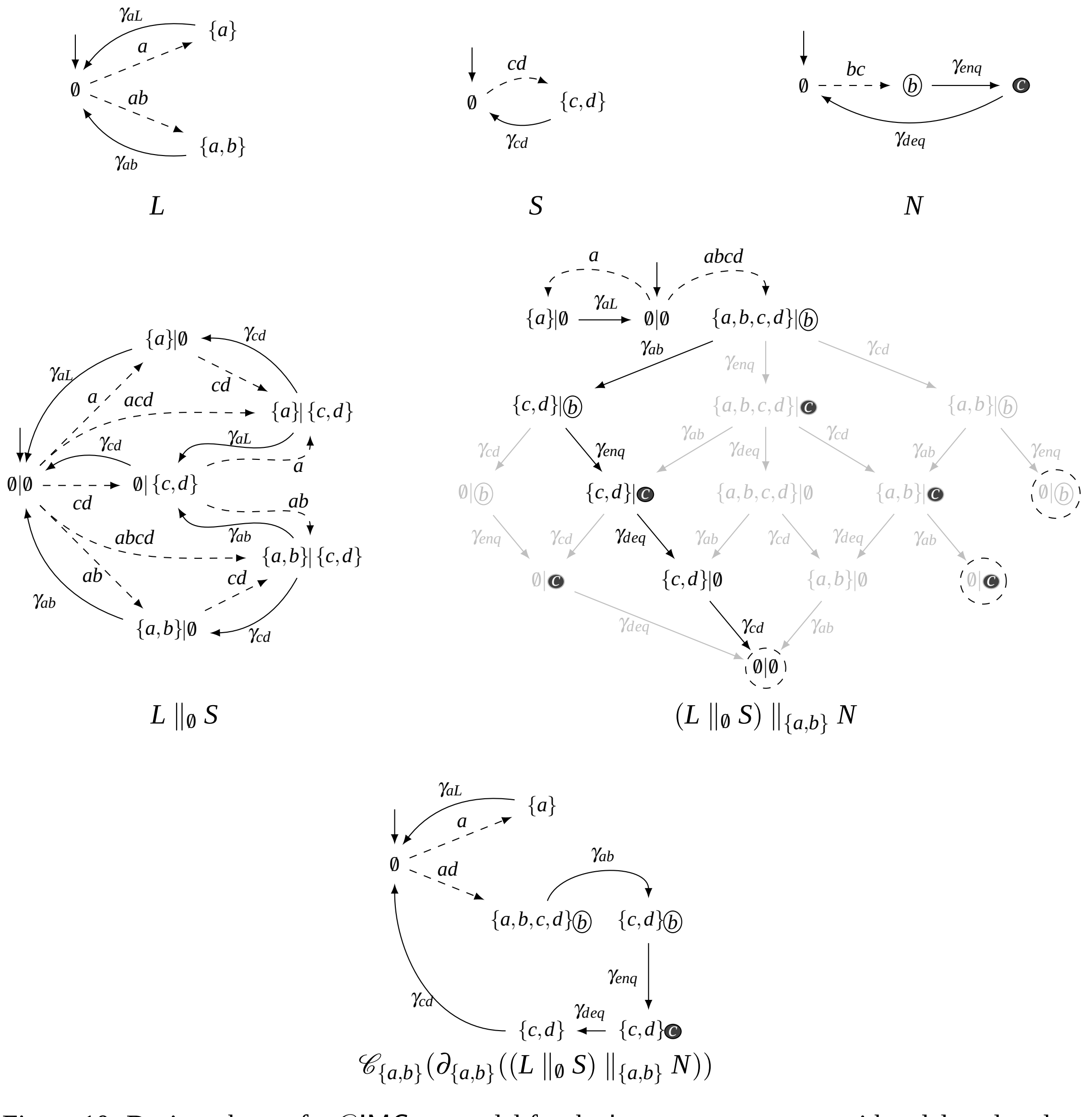


Figure 10: Design-phase of a $\mathscr{D}$IMC$_{\mathsf{Reo}}$ model for the lossysync connector with a delayed node.

In order to obtain the deployment-phase model, an extra step is required that composes the design-phase model with the environment model. Formally,

Definition 3 (Deployment)**.** *Let I be a* $\mathscr{D}$IMC$_{\mathsf{Reo}}$ *model of a design-phase connector, E a set of* $\mathscr{D}$IMC$_{\mathsf{Reo}}$ *models representing all the relevant reader and writer components defining the environment for I, and* *finally $M \subseteq \mathcal{N}$. The deployment-phase model of I in environment E with respect to the set of ports M is computed by*

$$\mathscr{C}_M(I \parallel_M E_{\parallel}),$$

where $E_{\parallel}$ is the parallel composition of all elements of E, referred to as the global environment model.

Note that whenever nodes do not delay the system, composition of $\mathscr{D}\mathsf{IMC}_{\mathsf{Reo}}$ models are boiled down to that defined for $\mathsf{IMC}_{\mathsf{Reo}}$. This is stated formally in the following theorem proved in [12]:

Theorem 1. *Let I be an $\mathsf{IMC}_{\mathsf{Reo}}$ and J a deployed $\mathscr{D}\mathsf{IMC}_{\mathsf{Reo}}$. Consider that both I and J model the same Stochastic Reo connector (*i.e.* with same stochastic information for channels and environment). Then, $I \sim J$ iff J has no enqueuing and dequeueing states.*

5 Concluding

This paper introduced $\mathscr{D}\mathsf{IMC}_{\mathsf{Reo}}$— a model for Stochastic Reo based on interactive Markov chains, which extends our previous work on $\mathsf{IMC}_{\mathsf{Reo}}$, increasing its scalability while retaining expressivity and compositionality. We believe coordination models are a major area of application of formal models to cloud applications, with an enormous potential for their correct design and analysis.

This debate, however, is still in its infancy; only time and experience with real, challenging application, will provide sustainable evidence for the claim made here, as well as for the approach proposed.

References

[1] Farhad Arbab (2003): *Abstract Behavior Types: A Foundation Model for Components and Their Composition*. In Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf & Willem-Paul de Roever, editors: *Formal Methods for Components and Objects*, *Lecture Notes in Computer Science* 2852, Springer, pp. 33–70 doi:10.1007/978-3-540-39656-7_2.

[2] Farhad Arbab (2004): *Reo: a channel-based coordination model for component composition*. *Mathematical Structures in Computer Science* 14(3), pp. 329–366 doi:10.1017/S0960129504004153.

[3] Farhad Arbab, Tom Chothia, Rob van der Mei, Sun Meng, YoungJoo Moon & Chrétien Verhoef (2009): *From Coordination to Stochastic Models of QoS*. In John Field & Vasco Vasconcelos, editors: *Coordination Models and Languages*, *Lecture Notes in Computer Science* 5521, Springer, pp. 268–287 doi:10.1007/978-3-642-02053-7_14

[4] Adnan Aziz, Kumud Sanwal, Vigyan Singhal & Robert Brayton (2000): *Model-checking continuous-time Markov chains*. *Transactions on Computational Logic* 1, pp. 162–170 doi:10.1145/343369.343402.

[5] Jos C. M. Baeten (2005): *A brief history of process algebra*. *Theoretical Computer Science* 335(2-3), pp. 131–146 doi:10.1016/j.tcs.2004.07.036.

[6] Christel Baier, Boudewijn Haverkort, Holger Hermanns & Joost P. Katoen (2003): *Model-Checking Algorithms for Continuous-Time Markov Chains*. *IEEE Transactions on Software Engineering* 29(6), pp. 524–541 doi:10.1109/TSE.2003.1205180.

[7] Marcello M. Bonsangue, Dave Clarke & Alexandra Silva (2012): *A model of context-dependent component connectors*. *Science of Computer Programming* 77(6), pp. 685–706 doi:10.1016/j.scico.2011.01.006.

[8] D. Gelernter & N. Carrier (1992): *Coordination Languages and their significance*. *Communication of the ACM* 2(35), pp. 97–107 doi:10.1145/129630.129635.

[9] Holger Hermanns (2002): *Interactive Markov Chains: The Quest for Quantified Quality*. *Lecture Notes in Computer Science* 2428, Springer, Berlin, Heidelberg doi:10.1007/3-540-45804-2_7.

[10] Young-Joo Moon (2011): *Stochastic Models for Quality of Service of Component Connectors*. Ph.D. thesis, Universiteit Leiden.

[11] Young-Joo Moon, Alexandra Silva, Christian Krause & Farhad Arbab (2014): *A compositional model to reason about end-to-end QoS in Stochastic Reo connectors*. *Science of Computer Programming* 80, pp. 3–24 doi:10.1016/j.scico.2011.11.007.

[12] Nuno Oliveira (2015): *Architectural reconfiguration of interacting services*. Ph.D. thesis, Universidades do Minho, Aveiro and Porto (Joint MAP-i Doctoral Programme).

[13] Nuno Oliveira, Alexandra Silva & Luis S. Barbosa (2015): $\mathsf{IMC_{Reo}}$*: interactive Markov chains for stochastic Reo*. *Journal of Internet Services and Information Security* 5(1), pp. 3–28.

Appendix - Composition in $\mathsf{IMC_{Reo}}$

In [13] the composition of two $\mathsf{IMC_{Reo}}$ models I_1 and I_2, with respect to $M \subseteq \mathcal{N}$, is given by

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

comprising a *product* and a *synchronization* operator. This appendix recalls the corresponding definition.

Definition 4 (Parallel Composition). *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 $\mathsf{IMC_{Reo}}$ models. The parallel composition of I and J with respect to a set $M \subseteq \mathcal{N}$ is defined as*

$$I \;\|_M\; J = (S, Act, \dashrightarrow, \longrightarrow, (s_i, s_j))$$

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

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

The first three clauses in Definition 4 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.

Definition 5 (Synchronisation). *Let $I = (S_1 \times S_2, Act, \dashrightarrow, \longrightarrow, s)$ be an $\mathsf{IMC_{Reo}}$ model over a composite state space, and $M \subseteq \mathcal{N}$. 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 $S_M = \{(i, j)\restriction_M \mid (i, j) \in S_1 \times S_2\}$ and $\dashrightarrow_M$ and $\longrightarrow_M$ are the smallest relations satisfying, respectively, conditions 1 and 2 below:

1. $$\frac{(i, j) \overset{X}{\dashrightarrow} (i', j') \quad (i, j) \not\triangleright M}{(i, j)\restriction_M \overset{X \setminus M}{\dashrightarrow}_M (i', j')\restriction_M}$$
2. $$\frac{(i, j) \xrightarrow{\gamma} (i', j') \quad (R_{i'} \cup R_{j'}) \cap M = \emptyset}{(i, j)\restriction_M \xrightarrow{\gamma}_M (i', j')\restriction_M}$$