Stochastic models for quality of service of component connectors

Moon, Y.J.

Chapter 4

Stochastic Reo Automata

4.1 Introduction

In the previous chapter, we introduced Quantitative Intentional Automata (QIA), a compositional semantic model for Stochastic Reo. QIA specify the behavior of connectors and enable reasoning about their end-to-end QoS. However, QIA explicitly describe all I/O interaction with the environment which is abstracted away in other (non-stochastic) semantic models such as Constraint Automata (CA) and Reo Automata. An explicit description of all interaction with the environment produces many states and transitions. Having a large state diagram, QIA are not easy to handle.

In this chapter, we introduce Stochastic Reo Automata [68] as an alternative semantic model for Stochastic Reo. Not only a Stochastic Reo Automaton is compact and tractable, but it also retains the features of QIA for representing context-dependency and reasoning about end-to-end QoS. Moreover, in order to reason about general end-to-end QoS, a Stochastic Reo Automaton is extended with reward information to deal with Stochastic Reo with rewards.

This chapter consists of four parts. In the first part, Stochastic Reo Automata are introduced as an alternative semantic model for Stochastic Reo. In fact, this model is a stochastic extension of Reo Automata. We introduce that the mapping between primitive Reo channels and their corresponding Stochastic Reo Automata, as well as the composition operation for Stochastic Reo Automata.

The second part shows the extended version of Stochastic Reo Automata for general end-to-end QoS properties. In this extension, the general QoS aspects are considered as reward information, which is associated with stochastic activities such as I/O request arrivals at channel ends and data-flows through channels. We also show the mapping of Stochastic Reo to Stochastic Reo Automata with the concern for the reward information.

The third part shows the translation from Stochastic Reo Automata into homogeneous CTMCs. In Chapter 3, we have shown the translation from QIA into CTMCs. This translation is partially similar to the translation from Stochastic Reo Automata into CTMCs. To avoid duplication, we skip some procedures that we reuse from the

earlier translation method. In addition, we present the translation from the Stochastic Reo Automata extended with reward information into CTMCs with state reward.

In the fourth part, we discuss to what extent Interactive Markov Chains (IMCs) can serve as another semantic model for Stochastic Reo. As shown in Section 2.5, the main strength of IMCs is their compositionality. In this section, we show that in our treatment the compositionality of IMCs is not adequate to specify the behavior of Stochastic Reo.

4.2 Stochastic Reo Automata

Stochastic Reo Automata constitute an alternative semantic model for Stochastic Reo to the model explained in Chapter 3. Compared to QIA, each Stochastic Reo Automaton has a disjoint set of node names. For example, two QIA $\mathcal{A}_1$ and $\mathcal{A}_2$ with node sets $\Sigma_{\mathcal{A}_1}$ and $\Sigma_{\mathcal{A}_2}$, respectively, are synchronized at nodes in $\Sigma_{\mathcal{A}_1} \cap \Sigma_{\mathcal{A}_2}$, whereas two Stochastic Reo Automata $\mathcal{B}_1$ and $\mathcal{B}_2$ are assumed that the two automata have disjoint node sets and can either take a step together or independently. Thus, naming the nodes of Stochastic Reo for a Stochastic Reo Automaton is slightly different from that for QIA. For instance, compared to Figure 2.5, Figure 4.1 shows the difference for the primitive channels of a LossySync and a FIFO1 and their composition result, i.e., the joined nodes in Figure 4.1 do not use a common name.

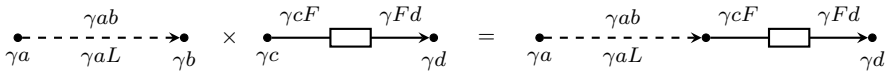


Figure 4.1: Stochastic LossyFIFO1 connector

As a more complex Stochastic Reo connector, Figure 4.2 shows a *discriminator* which takes the first arriving input value and produces it as its output. It also ensures that an input value arrives on every other input node before the next round.

4.2.1 Stochastic Reo Automata

In this section, we provide a compositional semantics for Stochastic Reo connectors, as an extension of the Reo Automata of Section 2.3.3 with functions that assign stochastic values for data-flows and I/O request arrivals.

Definition 4.2.1 (Stochastic Reo Automata). *A Stochastic Reo Automaton is a triple* $(\mathcal{A}, \mathbf{r}, \mathbf{t})$ *with a Reo Automaton* $\mathcal{A} = (\Sigma, Q, \delta_{\mathcal{A}})$ *according to Definition 2.3.7 and*

- $\mathbf{r} : \Sigma \to \mathbb{R}^+$ *is a function that associates with each node its arrival rate.*
- $\mathbf{t} : \delta_{\mathcal{A}} \to 2^{\Theta}$ *is a function that associates with a transition a subset of* $\Theta = 2^{\Sigma} \times 2^{\Sigma} \times \mathbb{R}^+$ *such that for any* $I, O \subseteq \Sigma$ *and* $I \cap O = \emptyset$*, each* $(I, O, r) \in \Theta$

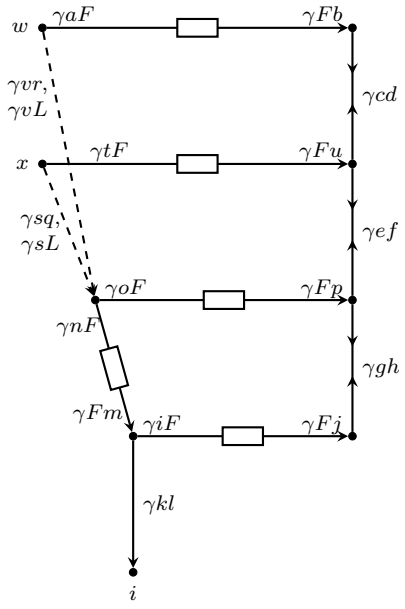


Figure 4.2: Stochastic Discriminator with two inputs

corresponds to a data-flow where I is a set of mixed and/or input nodes; O is a set of output and/or mixed nodes; and r is a processing delay rate for the data-flow described by I and O. We require that

- *for any two 3-tuples (I_1, O_1, r_1), $(I_2, O_2, r_2) \in \Theta$ such that $I_1 = I_2 \wedge O_1 = O_2$, it holds that $r_1 = r_2$, and*
- *for a transition $s \xrightarrow{g|f} s' \in \delta_{\mathcal{A}}$ with $\mathbf{t}(s \xrightarrow{g|f} s') = \{(I_1, O_1, r_1), (I_2, O_2, r_2), \ldots, (I_n, O_n, r_n)\}$, $f \setminus (I \cap O) = (I \cup O) \setminus (I \cap O)$ where $I = \bigcup_{1 \leq i \leq n} I_i$ and $O = \bigcup_{1 \leq i \leq n} O_i$.*

■

The Stochastic Reo Automata corresponding to the primitive Stochastic Reo channels in Figure 2.3 are defined by the functions $\mathbf{r}$ and $\mathbf{t}$ shown in Table 4.1. Note that the function $\mathbf{t}$ is encoded in the labels of the transitions of the automata, and the function $\mathbf{r}$ is shown inside the tables. For simplicity, here and in the remainder of this chapter, we simplify the representation of the 3-tuple (I, O, r), which is assigned by the function $\mathbf{t}$, by omitting the curly brackets for I and O and the commas between the elements in I and O.

An element of $\theta \in \Theta$ is accessed by projection functions $i : \Theta \to 2^{\Sigma}$, $o : \Theta \to 2^{\Sigma}$ and $v : \Theta \to \mathbb{R}^{+}$; $i(\theta)$ and $o(\theta)$ return the respective input and output nodes of

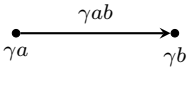
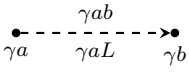
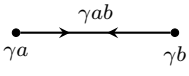
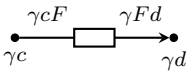
Synchronous Channels		
	$ab\|ab,\ \{(a,b,\gamma ab)\}$ 	$\begin{array}{\|c\|c\|}\hline & \mathbf{r} \\ \hline a & \gamma a \\ b & \gamma b \\ \hline \end{array}$
	$ab\|ab,\ \{(a,b,\gamma ab)\}$ $a\bar{b}\|a,\ \{(a,\emptyset,\gamma aL)\}$ 	$\begin{array}{\|c\|c\|}\hline & \mathbf{r} \\ \hline a & \gamma a \\ b & \gamma b \\ \hline \end{array}$
	$ab\|ab,\ \{(ab,\emptyset,\gamma ab)\}$ 	$\begin{array}{\|c\|c\|}\hline & \mathbf{r} \\ \hline a & \gamma a \\ b & \gamma b \\ \hline \end{array}$
Asynchronous Channel		
	$c\|c,\ \{(c,\emptyset,\gamma cF)\}$  $d\|d,\ \{(\emptyset,d,\gamma Fd)\}$	$\begin{array}{\|c\|c\|}\hline & \mathbf{r} \\ \hline c & \gamma c \\ d & \gamma d \\ \hline \end{array}$

Table 4.1: Stochastic Reo Automaton for some basic Stochastic Reo channels

a data-flow, and $v(\theta)$ returns the delay rate of the data-flow through nodes in $i(\theta)$ and $o(\theta)$.

As mentioned in Section 2.3.3, Reo Automata provide a compositional semantics for Reo connectors. As an extension of Reo Automata, Stochastic Reo Automata also present the composition of Stochastic Reo connectors at the automata level. For this purpose, we define the two operations of product and synchronization that are used to obtain an automaton of a Stochastic Reo connector by composing the automata of its primitive connectors. The compositionality of these operations is formally proved later in this section.

Definition 4.2.2 (Product). *Given two Stochastic Reo Automata* $(\mathcal{A}_1, \mathbf{r}_1, \mathbf{t}_1)$ *and* $(\mathcal{A}_2, \mathbf{r}_2, \mathbf{t}_2)$ *with* $\mathcal{A}_1 = (\Sigma_1, Q_1, \delta_{\mathcal{A}_1})$ *and* $\mathcal{A}_2 = (\Sigma_2, Q_2, \delta_{\mathcal{A}_2})$*, their product* $(\mathcal{A}_1, \mathbf{r}_1, \mathbf{t}_1) \times (\mathcal{A}_2, \mathbf{r}_2, \mathbf{t}_2)$ *is defined as* $(\mathcal{A}_1 \times \mathcal{A}_2, \mathbf{r}_1 \cup \mathbf{r}_2, \mathbf{t})$ *where*

$$
\begin{array}{lcll}
\mathbf{t}((q,p) \xrightarrow{gg'|ff'} (q',p')) & = & \mathbf{t}_1(q \xrightarrow{g|f} q') \cup \mathbf{t}_2(p \xrightarrow{g'|f'} p') & \\
 & & \multicolumn{2}{l}{\textit{where } q \xrightarrow{g|f} q' \in \delta_1 \wedge p \xrightarrow{g'|f'} p' \in \delta_2} \\
\mathbf{t}((q,p) \xrightarrow{gp^\sharp|f} (q',p)) & = & \mathbf{t}_1(q \xrightarrow{g|f} q') & \textit{where } q \xrightarrow{g|f} q' \in \delta_1 \wedge p \in Q_2 \\
\mathbf{t}((q,p) \xrightarrow{gq^\sharp|f} (q,p')) & = & \mathbf{t}_2(p \xrightarrow{g|f} p') & \textit{where } p \xrightarrow{g|f} p' \in \delta_2 \wedge q \in Q_1
\end{array}
$$

■

Note that we use $\times$ to denote both the product of Reo Automata and the product of Stochastic Reo Automata.

The set of 3-tuples that $\mathbf{t}$ associates with a transition m combines the delay rates involved in all data-flows synchronized by the transition m. For this combining, we might use a representative value for the synchronized data-flows, for example, the maximum of the delay rates. However, deciding the representative rate is not always desirable, and moreover, it can cause restriction to modeling random behavior of a system. In order to keep Stochastic Reo Automata generally useful and compositional, and their product commutative, we avoid fixing the precise formal meaning of distribution rates of synchronized transitions composed in a product. Instead, we represent the "delay rate" of a composite transition in the product automaton as the union of the delay rates of the synchronizing transitions of the two automata. The way these rates are combined to yield the composite rate of the transition depends on the different properties of the distributions and their time domains. For example, in the continuous-time case, no two events can occur at the same time; and we might choose the maximum one among the rates of the synchronized data-flows as their representative rate, but the exponential distributions are not closed under taking maximum. In Section 4.4, we show how to translate a Stochastic Reo Automaton to a CTMC using the union of the rates of the exponential distributions in the continuous-time case.

Definition 4.2.3 (Synchronization). *Given a Stochastic Reo Automaton* $(\mathcal{A}, \mathbf{r}, \mathbf{t})$ *with* $\mathcal{A} = (\Sigma, Q, \delta)$*, the synchronization operation for nodes* a *and* b *is defined as* $\partial_{a,b}(\mathcal{A}, \mathbf{r}, \mathbf{t}) = (\partial_{a,b}\mathcal{A}, \mathbf{r}', \mathbf{t}')$ *where*

- $\mathbf{r}'$ *is* $\mathbf{r}$ *restricted to the domain* $\Sigma \setminus \{a, b\}$.
- $\mathbf{t}'$ *is defined as:*

$$\mathbf{t}'(q \xrightarrow{g\setminus_{ab}|f\setminus\{a,b\}} q') = \{(A', B', r) \mid (A, B, r) \in \mathbf{t}(q \xrightarrow{g|f} q'), \\ A' = sync(A, \{a, b\}) \wedge \ B' = sync(B, \{a, b\}) \ \}$$

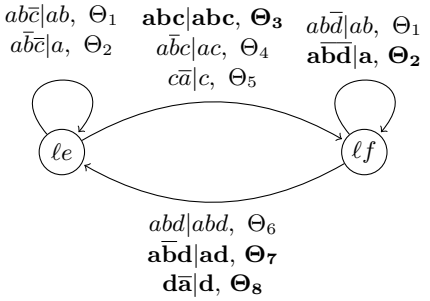
where $sync : 2^\Sigma \times 2^\Sigma \to 2^\Sigma$ *gathers nodes joined by synchronization, and is defined as:*

$$sync(A, B) = \begin{cases} A \cup B & \text{if } A \cap B \neq \emptyset \\ A & \text{otherwise} \end{cases}$$

■

Note that we use the symbol $\partial_{a,b}$ to denote both the synchronization of Reo Automata and the synchronization of Stochastic Reo Automata. The number of nodes joined by a synchronization is always two, and these joined nodes are gathered in a one set. The sets of joined nodes in multiple synchronization steps are disjoint. That is, given two different synchronizations $\partial_{a,b}$ and $\partial_{c,d}$ on a Stochastic Reo automaton, $\{a, b\} \cap \{c, d\} = \emptyset$.

Example 4.2.4. We now revisit the LossyFIFO1 example. Its semantics is given by the triple $(\mathcal{A}_{LossyFIFO1}, \mathbf{r}, \mathbf{t})$, where $\mathcal{A}_{LossyFIFO1}$ is the automaton depicted in Figure 2.11 and $\mathbf{r}$ is defined as $\mathbf{r} = \{a \mapsto \gamma a, b \mapsto \gamma b, c \mapsto \gamma c, d \mapsto \gamma d\}$. For $\mathbf{t}$, we first compute $\mathbf{t}_{LossySync \times FIFO1}$:



Θ_1 :	$\{(a, b, \gamma ab)\}$
Θ_2 :	$\{(a, \emptyset, \gamma aL)\}$
Θ_3 :	$\{(a, b, \gamma ab), (c, \emptyset, \gamma cF)\}$
Θ_4 :	$\{(a, \emptyset, \gamma aL), (c, \emptyset, \gamma cF)\}$
Θ_5 :	$\{(c, \emptyset, \gamma cF)\}$
Θ_6 :	$\{(a, b, \gamma ab), (\emptyset, d, \gamma Fd)\}$
Θ_7 :	$\{(a, \emptyset, \gamma aL), (\emptyset, d, \gamma Fd)\}$
Θ_8 :	$\{(\emptyset, d, \gamma Fd)\}$

Above, the labels that correspond to the transitions that will be kept after synchronization appear in **bold**. Thus, the result of joining nodes by synchronization, is shown in Figure 4.3 with $\mathbf{r}' = \{a \mapsto \gamma a, d \mapsto \gamma d\}$ which is restricted to its boundary nodes.

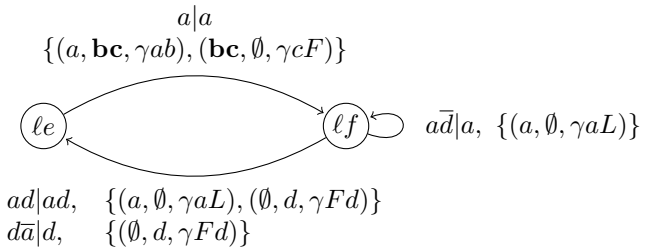


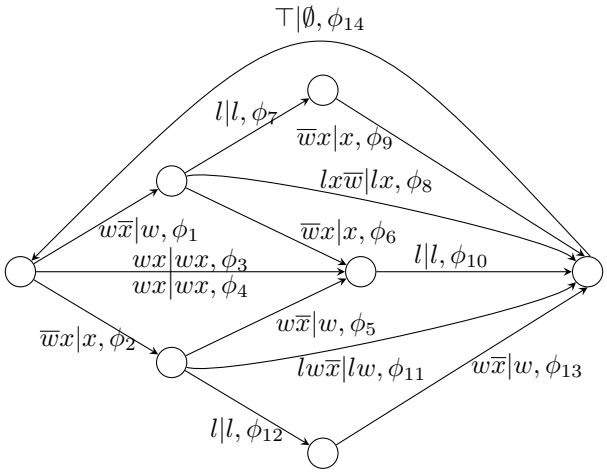
Figure 4.3: Stochastic Reo Automaton for LossyFIFO1

Note that the node names that appear in **bold** represent the synchronization of nodes b and c. For simplicity, here and in the remainder of this chapter, we use abbreviated representation of states, for example, the states ℓe and ℓf in the above automaton are abbreviations for (ℓ, e) and (ℓ, f). ◊

In this way, we can carry on stochastic information, i.e., arrival rates and processing delay rates that pertain to its QoS, in the semantic model of Reo circuits, given as Reo Automata.

As a more complex example of such composition, Figure 4.4 shows a Stochastic Reo Automaton for the discriminator in Figure 4.2.

Definition 4.2.1 shows that our extension of Reo Automata deals with such stochastic information separately, apart from the underlying Reo Automaton. Thus, our extended model retains the properties of Reo Automata, i.e., the compositionality result presented in Section 2.3.3 can be extended to Stochastic Reo Automata.



$$
\begin{array}{lcll}
\phi_1 & = & \{ & (w, av, \gamma avw),\ (av, r, \gamma vr),\ (r, no, \gamma nor), \\
 & & & \quad (no, \emptyset, \gamma oF),\ (no, \emptyset, \gamma nF),\ (av, \emptyset, \gamma aF)\ \} \\
\phi_2 & = & \{ & (x, st, \gamma stx),\ (st, \emptyset, \gamma tF),\ (st, q, \gamma sq), \\
 & & & (q, no, \gamma noq),\ (no, \emptyset, \gamma oF),\ (no, \emptyset, \gamma nF)\ \} \\
\phi_3 & = & & \phi_2 \cup \phi_5 \\
\phi_4 & = & & \phi_1 \cup \phi_6 \\
\phi_5 & = & \{ & (w, av, \gamma avw),\ (av, \emptyset, \gamma vL),\ (av, \emptyset, \gamma aF)\ \} \\
\phi_6 & = & \{ & (x, st, \gamma stx)\ , (st, \emptyset, \gamma tF),\ (st, \emptyset, \gamma sL)\ \} \\
\phi_7 & = & \{ & (\emptyset, m, \gamma Fm),\ (m, ik, \gamma ikm),\ (ik, l, \gamma kl), \\
 & & & (ik, \emptyset, \gamma iF)\ \} \\
\phi_8 & = & & \phi_7 \cup \phi_6 \\
\phi_9 & = & & \phi_6 \\
\phi_{10} & = & & \phi_{12}\ =\ \phi_7 \\
\phi_{11} & = & & \phi_7 \cup \phi_5 \\
\phi_{13} & = & & \phi_5 \\
\phi_{14} & = & \{ & (\emptyset, bc, \gamma Fb),\ (bcde, \emptyset, \gamma cd),\ (\emptyset, u, \gamma Fu) \\
 & & & (u, de, \gamma deu),\ (\emptyset, p, \gamma Fp),\ (defg, \emptyset, \gamma ef), \\
 & & & (p, fg, \gamma fgp),\ (fghj, \emptyset, \gamma gh), (\emptyset, hj, \gamma Fj)\ \}
\end{array}
$$

Figure 4.4: Stochastic Reo Automaton for discriminator in Figure 4.2

Given two Stochastic Reo Automata $(\mathcal{A}_1, \mathbf{r}_1, \mathbf{t}_1)$ and $(\mathcal{A}_2, \mathbf{r}_2, \mathbf{t}_2)$ with $\mathcal{A}_1 = (\Sigma_1, Q_1, \delta_1)$ and $\mathcal{A}_2 = (\Sigma_2, Q_2, \delta_2)$ over the disjoint alphabets Σ_1 and Σ_2, and two subsets $\{a_1, \ldots, a_k\} \subseteq \Sigma_1$ and $\{b_1, \ldots, b_k\} \subseteq \Sigma_2$, we construct $\partial_{a_1,b_1}\partial_{a_2,b_2}\cdots\partial_{a_k,b_k}(\mathcal{A}_1 \times \mathcal{A}_2)$ as the automaton corresponding to a connector where node a_i of the first connector is connected to node b_i of the second connector, for all $i \in \{1, \ldots, k\}$. Note that the

'plugging' order does not matter because ∂ can be applied in any order and it interacts well with the product. These properties are captured in the following lemma.

Lemma 4.2.5 (Compositionality). *Given two disjoint Stochastic Reo Automata* $(\mathcal{A}_1, \mathbf{r}_1, \mathbf{t}_1)$ *and* $(\mathcal{A}_2, \mathbf{r}_2, \mathbf{t}_2)$ *with* $\mathcal{A}_1 = (\Sigma_1, Q_1, \delta_1)$ *and* $\mathcal{A}_2 = (\Sigma_2, Q_2, \delta_2)$,

1. $\partial_{a,b}\partial_{c,d}(\mathcal{A}_1, \mathbf{r}_1, \mathbf{t}_1) = \partial_{c,d}\partial_{a,b}(\mathcal{A}_1, \mathbf{r}_1, \mathbf{t}_1)$, *if* $a, b, c, d \in \Sigma_1$
2. $(\partial_{a,b}(\mathcal{A}_1, \mathbf{r}_1, \mathbf{t}_1)) \times (\mathcal{A}_2, \mathbf{r}_2, \mathbf{t}_2) \sim \partial_{a,b}((\mathcal{A}_1, \mathbf{r}_1, \mathbf{t}_1) \times (\mathcal{A}_2, \mathbf{r}_2, \mathbf{t}_2))$, *if* $a, b \notin \Sigma_2$

Here $(\mathcal{A}_1, \mathbf{r}_1, \mathbf{t}_1) \sim (\mathcal{A}_2, \mathbf{r}_2, \mathbf{t}_2)$, *where* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are automata over the same alphabet, if and only if* $\mathcal{A}_1 \sim \mathcal{A}_2$, $\mathbf{r}_1 = \mathbf{r}_2$, *and* $\mathbf{t}_1 = \mathbf{t}_2$. ♦

Proof. For the first proposition, let

- $\partial_{a,b}\partial_{c,d}(\mathcal{A}_1, \mathbf{r}_1, \mathbf{t}_1) = (\partial_{a,b}\partial_{c,d}\mathcal{A}_1, \mathbf{r}'_1, \mathbf{t}'_1)$ and
- $\partial_{c,d}\partial_{a,b}(\mathcal{A}_1, \mathbf{r}_1, \mathbf{t}_1) = (\partial_{c,d}\partial_{a,b}\mathcal{A}_1, \mathbf{r}''_1, \mathbf{t}''_1)$

By [19, Lemma 4.13] which is the analogue result for Reo Automata, we know that $\partial_{a,b}\partial_{c,d}\mathcal{A}_1 = \partial_{c,d}\partial_{a,b}\mathcal{A}_1$. Using basic set theory, we also have that

$$\begin{array}{rcl} \mathbf{r}'_1 & = & \mathbf{r} \mid (\Sigma \setminus \{a,b\}) \setminus \{c,d\} \\ & = & \mathbf{r} \mid (\Sigma \setminus \{c,d\}) \setminus \{a,b\} \\ & = & \mathbf{r}''_1 \end{array}$$

where for $x \subseteq \Sigma$, $\mathbf{r}|x$ is the restriction of $\mathbf{r}$ to x.

Before moving to the fact that $\mathbf{t}'_1 = \mathbf{t}''_1$, we show that the order of applying the synchronization is irrelevant for the synchronization result, i.e., given three node sets A, $\{a,b\}$, and $\{c,d\}$, and the synchronization function in Definition 4.2.3,

$$sync(sync(A, \{a,b\}), \{c,d\}) = sync(sync(A, \{c,d\}), \{a,b\})$$

because, given three node sets A, B, and C with $B \cap C = \emptyset$,

$$sync(sync(A,B),C) = \begin{cases} A \cup B \cup C & \text{if } A \cap B \neq \emptyset \ \wedge \ A \cap C \neq \emptyset \\ A \cup B & \text{if } A \cap B \neq \emptyset \ \wedge \ A \cap C = \emptyset \\ A \cup C & \text{if } A \cap B = \emptyset \ \wedge \ A \cap C \neq \emptyset \\ A & \text{otherwise} \end{cases}$$

and set union $\cup$ is commutative. Now

$$\begin{array}{rl} & \mathbf{t}'_1(q \xrightarrow{g \setminus abcd \mid (f \setminus \{a,b\}) \setminus \{c,d\}} q') \\ = & \{(A', B', r) \mid (A, B, r) \in \mathbf{t}_1(q \xrightarrow{g|f} q'), \\ & \qquad A'' = sync(A, \{a,b\}) \ \wedge \ B'' = sync(B, \{a,b\}) \ \wedge \\ & \qquad A' = sync(A'', \{c,d\}) \ \wedge \ B' = sync(B'', \{c,d\})\} \\ = & \{(A', B', r) \mid (A, B, r) \in \mathbf{t}_1(q \xrightarrow{g|f} q'), \\ & \qquad A'' = sync(A, \{c,d\}) \ \wedge \ B'' = sync(B, \{c,d\}) \ \wedge \\ & \qquad A' = sync(A'', \{a,b\}) \ \wedge \ B' = sync(B'', \{a,b\})\} \\ = & \mathbf{t_1}''(q \xrightarrow{g \setminus cdab \mid (f \setminus \{c,d\}) \setminus \{a,b\}} q') \end{array}$$

For the second proposition, let

- $\partial_{a,b}(\mathcal{A}_1, \mathbf{r}_1, \mathbf{t}_1) \times (\mathcal{A}_2, \mathbf{r}_2, \mathbf{t}_2) = (\partial_{a,b}(\mathcal{A}_1) \times \mathcal{A}_2, \mathbf{r}, \mathbf{t})$ and
- $\partial_{a,b}((\mathcal{A}_1, \mathbf{r}_1, \mathbf{t}_2) \times (\mathcal{A}_2, \mathbf{r}_2, \mathbf{t}_2)) = (\partial_{a,b}(\mathcal{A}_1 \times \mathcal{A}_2), \mathbf{r}', \mathbf{t}')$

By [19, Lemma 4.13], we know that $\partial_{a,b}(\mathcal{A}_1) \times \mathcal{A}_2 \sim \partial_{a,b}(\mathcal{A}_1 \times \mathcal{A}_2)$ if $a, b \notin \Sigma_2$. It remains to prove that $\mathbf{r} = \mathbf{r}'$ and $\mathbf{t} = \mathbf{t}'$.

For the first part, we easily calculate:

$$\mathbf{r}(p) \quad = \quad \begin{cases} \mathbf{r}_1(p) & \text{if } \ p \in \Sigma_1 \setminus \{a,b\} \\ \mathbf{r}_2(p) & \text{if } \ p \in \Sigma_2 \end{cases} \quad = \quad \mathbf{r}'(p)$$

For the second part, consider transitions $(q_1, q_2) \xrightarrow{(g_1 \setminus ab) g_2 | (f_1 \setminus \{a,b\}) f_2} (p_1, p_2)$ in $\partial_{a,b}(\mathcal{A}_1) \times \mathcal{A}_2$ and $(q_1, q_2) \xrightarrow{(g_1 g_2) \setminus ab | (f_1 f_2) \setminus \{a,b\}} (p_1, p_2)$ in $\partial_{a,b}(\mathcal{A}_1 \times \mathcal{A}_2)$ with $g_i \in \mathcal{B}_{\Sigma_i}$ and $f_i \in 2^{\Sigma_i}$ for $i = 1, 2$, which includes joined nodes a and b. Then

$$\begin{array}{lll}
& \mathbf{t}((q_1, q_2) \xrightarrow{(g_1 \setminus ab) g_2 | (f_1 \setminus \{a,b\}) f_2} (p_1, p_2)) & \\
= & \{(A', B', r) \mid (A, B, r) \in \mathbf{t}_1(q_1 \xrightarrow{g_1 | f_1} p_1), & \\
& \qquad\qquad A' = sync(A, \{a,b\}) \ \wedge \ B' = sync(B, \{a,b\})\} & \\
\cup & \{(A, B, r) \mid (A, B, r) \in \mathbf{t}_2(q_2 \xrightarrow{g_2 | f_2} p_2)\} & \\
= & \{(A', B', r) \mid (A, B, r) \in \mathbf{t}_1(q_1 \xrightarrow{g_1 | f_1} p_1) \ \cup \ \mathbf{t}_2(q_2 \xrightarrow{g_2 | f_2} p_2), & \\
& \qquad\qquad A' = sync(A, \{a,b\}) \ \wedge \ B' = sync(B, \{a,b\})\} & \\
= & \mathbf{t}'((q_1, q_2) \xrightarrow{(g_1 g_2) \setminus ab | (f_1 f_2) \setminus \{a,b\}} (p_1, p_2)) &
\end{array}$$

Since $sync(C, D) = C$ if $C \cap D = \emptyset$, the above result holds without a need to consider if $ab \leq g_1$ or $\{a, b\} \subseteq f_1$. This also implies that $\mathbf{t} = \mathbf{t}'$ holds for transitions $(q, p) \xrightarrow{g_1 | f_1} (q', p)$ and $(q, p) \xrightarrow{g_2 | f_2} (q, p')$, which do not include joined nodes, in $\partial_{a,b}(\mathcal{A}_1) \times \mathcal{A}_2$ (equivalently, in $\partial_{a,b}(\mathcal{A}_1 \times \mathcal{A}_2)$). □

4.3 Reward model

The end-to-end QoS aspects considered in Stochastic Reo typically involve timing information. Stochastic Reo is, however, general enough to include other types of quantitative information and, in this section, we consider rewards to model, for instance, CPU computation time, memory space, etc. Rewards are assigned to request-arrivals or data-flows in Stochastic Reo. Assigning a reward is done in a similar fashion to annotating an activity with a stochastic rate. This similarity is leading in the following section which discusses the extension of Stochastic Reo Automata with reward information.

4.3.1 Stochastic Reo with reward information

To specify the rewards of the behavior of Reo connectors, we have extended Stochastic Reo by associating reward information to the stochastic activities of request-arrivals at channel ends and data-flows through channels. Intuitively, the reward information indicates the amount of resources required or released (gained or lost) for carrying out the relevant stochastic activities.

In our extension, reward information is not confined to specific types. Moreover, multiple types of rewards can be associated with a single stochastic activity. The type of each reward is labeled to its reward value, for instance, *memory space: 3*. Formally, the reward information is an element of $(\mathit{Types} \times \mathbb{R})^*$ where *Types* is a set of reward types. For simplicity, here and throughout of this thesis, we do not explicitly mention reward types and assume that they are implied by the positions of the values in each sequence. Thus, we use $\mathbb{R}^*$ instead of $(\mathit{Types} \times \mathbb{R})^*$, where $\mathbb{R}^*$ is a sequence of real numbers, which we shall call a *reward sequence*. Let π be a reward sequence. The ith element of π is denoted by $\pi(i)$ and the length of π, by $|\pi|$. Implicitly, each $\pi(i)$ is associated with a certain type of reward.

Figure 4.5 shows some primitive Stochastic Reo channels with stochastic rates and reward sequences for stochastic activities. We associate a pair of a reward sequence and a stochastic rate with each of their relevant stochastic activities, represented in the format (*rate* | *reward sequence*).

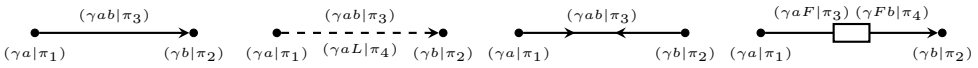


Figure 4.5: Some basic Stochastic Reo channels with rewards

For efficient reasoning about the rewards, we assume that all reward sequences of the same connector have the same length. For instance, for π_1, π_2, π_3 in the Sync channel in Figure 4.5, $|\pi_1| = |\pi_2| = |\pi_3|$. In addition, the reward values positioned at the same index in the reward sequences from the same connector must have the same type. For example, in the context of the FIFO1 channel in Figure 4.5: Let the reward types of π_3 associated with the data-flow from node a to the buffer be [*memory space*, *computation time*], then for the reward sequence π_4 associated with the data-flow from the buffer to node b, $|\pi_4| = 2$ and the order of reward types in π_4 must also be [*memory space*, *computation time*].

Stochastic Reo extended with reward information retains the compositionality of Stochastic Reo. As mentioned in Section 4.2, we assume that pumping data at mixed nodes is an immediate activity, thus, the rates of mixed nodes are considered as ∞. In the case of rewards, even if pumping data does not consume any time it still requires some rewards/resource to carry out this activity. We need to define how to determine reward sequences for pumping data at mixed nodes. For this purpose, we recall next the definition of a constraint semiring (c-semiring, for short).

Definition 4.3.1. ***(Constraint semiring [18])*** *A constraint semiring is a structure* $(C, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ *where* C *is a set,* $\mathbf{0}, \mathbf{1} \in C$*, and* $\oplus$ *and* $\otimes$ *are binary operations on* C *such that:*

- $\oplus$ *is commutative, associative, idempotent;* $\mathbf{0}$ *is its unit element, and* $\mathbf{1}$ *is its absorbing element.*
- $\otimes$ *is commutative, associative and distributes over* $\oplus$*;* $\mathbf{1}$ *is its unit element, and* $\mathbf{0}$ *is its absorbing element.*

■

It should be noted that the operation $\oplus$ induces a partial order $\trianglelefteq$ on C, which is defined by $c \trianglelefteq c'$ iff $c \oplus c' = c'$.

The c-semiring structure is appropriate when only one calculation is possible over its domain. In the case of Reo connectors, we need two different types of calculations: a sequential calculation through the connector for its overall reward and a parallel calculation for joining nodes into a mixed node. For these two different types of calculations, another algebraic structure, called Q-algebra, is used. We recall the definition of Q-algebra; here Q refers to the *QoS* or *quantitative* values.

Definition 4.3.2. ***(Q-algebra [29])*** *A Q-algebra is a structure* $\mathcal{R} = (C, \oplus, \otimes, \oplus, \mathbf{0}, \mathbf{1})$ *such that* $\mathcal{R}_{\otimes} = (C, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ *and* $\mathcal{R}_{\oplus} = (C, \oplus, \oplus, \mathbf{0}, \mathbf{1})$ *are both c-semirings.* C *is called* the domain of $\mathcal{R}$. ■

The set C is a set of reward values, and the operations $\otimes$ and $\oplus$ calculate rewards whose relevant activities occur sequentially and in parallel, respectively. That is, given $c_1,\ c_2 \in C$, $c_1 \otimes c_2$ is the composed reward of when c_2 follows c_1, and $c_1 \oplus c_2$ is the composed reward of when c_1 and c_2 occur concurrently. For instance, the Q-algebra for the shortest computation time can be given as $(\mathbb{R}^+ \cup \{\infty\}, min, +, +, \infty, 0)$.

The product of two Q-algebra is defined as:

Definition 4.3.3. ***(Product [29])*** *For two Q-algebras* $\mathcal{R}_1 = (C_1, \oplus_1, \otimes_1, \oplus_1, \mathbf{0}_1, \mathbf{1}_1)$ *and* $\mathcal{R}_2 = (C_2, \oplus_2, \otimes_2, \oplus_2, \mathbf{0}_2, \mathbf{1}_2)$*, their product is* $\mathcal{R}_1 \diamond \mathcal{R}_2 = \mathcal{R} = (C, \oplus, \otimes, \oplus, \mathbf{0}, \mathbf{1})$ *where*

- $C = C_1 \times C_2$
- $(c_1, c_2) \oplus (c_1', c_2') = (c_1 \oplus_1 c_1',\ c_2 \oplus_2 c_2')$
- $(c_1, c_2) \otimes (c_1', c_2') = (c_1 \otimes_1 c_1',\ c_2 \otimes_2 c_2')$
- $(c_1, c_2) \oplus (c_1', c_2') = (c_1 \oplus_1 c_1',\ c_2 \oplus_2 c_2')$
- $\mathbf{0} = (\mathbf{0}_1, \mathbf{0}_2)$
- $\mathbf{1} = (\mathbf{1}_1, \mathbf{1}_2)$

■

To deal with different types of rewards, we label them, as mentioned above, as elements of $(Types \times \mathbb{R})^*$, and use *a labeled Q-algebra*, which is defined as follows.

Definition 4.3.4. ***(A labeled Q-algebra [29])*** *For each $1 \leq i \leq n$, let $\mathcal{R}_i = (C_i, \oplus_i, \otimes_i, \oplus\!\!\!\cdot_i, \mathbf{0}_i, \mathbf{1}_i)$ be a Q-algebra. Associating distinct label l_i with each $\mathcal{R}_i$, denoted by $(l_i : \mathcal{R}_i)$, such that $l_i \neq l_j$ if $i \neq j$, the product of $\mathcal{R}_i$ is a labeled Q-algebra $\mathcal{R} = (C, \oplus, \otimes, \oplus\!\!\!\cdot, \mathbf{0}, \mathbf{1})$ if*

- $C = (\{l_1\} \times C_1) \times \cdots \times (\{l_n\} \times C_n)$
- $(l_1 : c_1, \ldots, l_n : c_n) \oplus (l_1 : c'_1, \ldots, l_n : c'_n) = (l_1 : (c_1 \oplus_1 c'_1), \ldots, l_n : (c_n \oplus_n c'_n))$
- $(l_1 : c_1, \ldots, l_n : c_n) \otimes (l_1 : c'_1, \ldots, l_n : c'_n) = (l_1 : (c_1 \otimes_1 c'_1), \ldots, l_n : (c_n \otimes_n c'_n))$
- $(l_1 : c_1, \ldots, l_n : c_n) \oplus\!\!\!\cdot (l_1 : c'_1, \ldots, l_n : c'_n) = (l_1 : (c_1 \oplus\!\!\!\cdot_1 c'_1), \ldots, l_n : (c_n \oplus\!\!\!\cdot_n c'_n))$
- $\mathbf{0} = (l_1 : \mathbf{0}_1, \ldots, l_n : \mathbf{0}_n)$
- $\mathbf{1} = (l_1 : \mathbf{1}_1, \ldots, l_n : \mathbf{1}_n)$

■

The product operation on Q-algebras in Definition 4.3.3 can be applied to labeled Q-algebras only if the order of the labels of two Q-algebras are identical.

Now we show how to comprise/obtain the reward for mixed nodes. When joining a sink node and a source node into a mixed node, the resulting reward for the mixed node is obtained using $\oplus\!\!\!\cdot$ on two rewards of each node since respective activities for each node occur in parallel. That is, let a mixed node be composed out of a sink node a and a source node b whose respective rewards are π_a and π_b, then the resulting reward for the mixed node is $\pi_a \oplus\!\!\!\cdot \pi_b$.

We now consider a merger and a replicator with reward information. As mentioned in Section 2.1, a merger selects its source node and dispenses a data item from the selected source node to its sink node. Consider the merger in Figure 4.6[1]. This merger has 3 source nodes a, b, and c and a sink node d whose respective rewards are π_a, π_b, π_c, and π_d, then the comprised reward for this merger is $(\pi_a \oplus \pi_b \oplus \pi_c) \otimes \pi_d$.

(a) merger (b) replicator

Figure 4.6: Magnified merger and replicator

[1]Note that for simplicity, a merger and a replicator are usually depicted as mixed nodes, but here we magnify them in order to explain how to calculate their rewards. However, the names of their source and sink nodes are omitted.

In the case of a replicator, it takes a data-item from its source node and dispenses the duplication of the data-item to its all sink nodes. Consider the replicator in Figure 4.6. It has a source node a and 3 sink nodes b, c, and d whose respective rewards are π_a, π_b, π_c, and π_d, then the comprised reward of this replicator is $\pi_a \otimes (\pi_b \oplus \pi_c \oplus \pi_d)$.

As an example for the composed reward of Reo connectors, Figure 4.7 depicts the Stochastic Reo extended with reward information for LossySync and FIFO1, together with the connector resulting from the composition of the two.

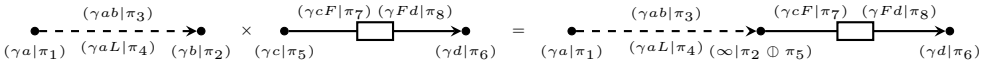


Figure 4.7: Stochastic Reo for LossyFIFO1 with rewards

Note that the $\times$ notation in Figure 4.7 represents joining the source node in a LossySync channel and the sink node in a FIFO1 channel.

4.3.2 Stochastic Reo Automata with reward information

As an operational semantic model for Stochastic Reo extended with reward information, we introduce an extended Stochastic Reo Automata model in this section (Definition 4.3.5). The reward information, which is described using reward sequences in Stochastic Reo, is propagated to the semantic model by pairing a stochastic rate with its relevant reward sequence.

Before moving to the definition of the semantic model for Stochastic Reo extended with reward information, we slightly modify extended Stochastic Reo to reuse the operations and the properties of Stochastic Reo Automata described in the previous sections. This modification is necessary to accommodate the rewards for mixed nodes. The original Stochastic Reo discards the rates for mixed nodes, but the extended Stochastic Reo explicitly represents them as ∞ to make a pair with the rewards for mixed nodes. In order to deal with this difference and reuse the methods for the original Stochastic Reo, an actual mixed node is replaced with an auxiliary Sync channel, and newly arising mixed nodes are considered as usual in the original Stochastic Reo. That is, the new mixed nodes are assumed not to consume any time and entail no rewards for pumping data. For compliance with this assumption, the LossyFIFO1 connector in Figure 4.7 is modified as follows:

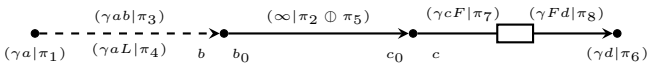


Figure 4.8: Modified LossyFIFO1 connector

Note that arbitrary names can be used for new mixed nodes in the modified con-

nectors, but here we use names as b_0 and c_0 to compare its corresponding automata model to the original Stochastic Reo Automaton corresponding to a LossyFIFO1 connector later. Adding an auxiliary Sync channel does not change the semantics of a connector and enables us to reuse the existing operations for Stochastic Reo Automata.

Definition 4.3.5. *[Stochastic Reo Automata extended with reward information] A Stochastic Reo Automaton with reward information is a tuple* $(\mathcal{A}, \mathbf{r}', \mathbf{t}', \mathcal{R})$ *where* $\mathcal{A} = (\Sigma, Q, \delta_{\mathcal{A}})$ *is a Reo Automaton and*

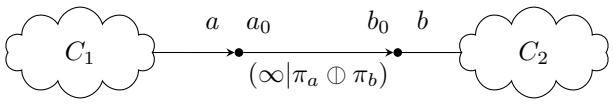
- $\mathbf{r}' : \Sigma \to \mathbb{R}^+ \times \mathbb{R}^*$ *is a function that associates with each node a pair consisting of an arrival rate and a reward sequence.*
- $\mathbf{t}' : \delta_{\mathcal{A}} \to 2^{\Psi}$ *is a function that associates with a transition* $q \xrightarrow{g|f} q' \in \delta_{\mathcal{A}}$ *a subset of* $\Psi = 2^{\Sigma} \times 2^{\Sigma} \times \mathbb{R}^+ \times \mathbb{R}^*$.
- $\mathcal{R}$*: a labeled Q-algebra* $(C, \oplus, \otimes, \oslash, \mathbf{0}, \mathbf{1})$ *with domain* C *of rewards.*

■

For each $\psi \in \Psi$, the projection functions that access its elements are $I : \Psi \to 2^{\Sigma}$, $O : \Psi \to 2^{\Sigma}$, $V : \Psi \to \mathbb{R}^+$, $R : \Psi \to \mathbb{R}^*$, and $pair : \Psi \to \mathbb{R}^+ \times \mathbb{R}^*$. I, O, and V return input nodes, output nodes, and its relevant rate, respectively, which correspond to i, o, and v in Section 4.2. R projects a relevant reward sequence from ψ. The function $pair$ returns a pair of a rate and its relevant reward sequence from ψ. We use $rate : \mathbb{R}^+ \times \mathbb{R}^* \to \mathbb{R}^+$ and $rew : \mathbb{R}^+ \times \mathbb{R}^* \to \mathbb{R}^*$ to access the elements of the results of the function $\mathbf{r}'$ and the function $pair$.

Table 4.2 shows Stochastic Reo Automata extended with reward information corresponding to the basic Stochastic Reo channels in Figure 4.5.

Now we show that using auxiliary Sync channels to retain rewards for mixed nodes does not affect the structure of Stochastic Reo Automata at all. Consider two connectors $\mathcal{C}_1$ and $\mathcal{C}_2$ with their boundary nodes, respectively, a and b, and these two connectors are connected by a Sync channel with ends a_0 and b_0 as follows:



Note that no rewards are assigned for the mixed nodes in the above connector, thus, we can reuse the definitions and the properties of the product and the synchronization for Stochastic Reo Automata, as mentioned in Section 4.2.

When the Stochastic Reo Automata extended with reward information for the connectors $\mathcal{C}_1$ and $\mathcal{C}_2$ are $(\mathcal{A}_1, \mathbf{r}'_1, \mathbf{t}'_1, \mathcal{R})$ with $\mathcal{A}_1 = (\Sigma_1, Q_1, \delta_1)$ and $(\mathcal{A}_2, \mathbf{r}'_2, \mathbf{t}'_2, \mathcal{R})$ with $\mathcal{A}_2 = (\Sigma_2, Q_2, \delta_2)$, respectively, the composition result of $\mathcal{C}_1$, $\mathcal{C}_2$, and the $\mathsf{Sync}(a_0, b_0)$ with $a_0,\ b_0 \notin \Sigma_1 \cup \Sigma_2$ is given by:

$$
\begin{aligned}
&(\partial_{b,b_0}((\partial_{a,a_0}(\mathcal{A}_1 \times \mathsf{Sync}(a_0, b_0))) \times \mathcal{A}_2), \mathbf{r}', \mathbf{t}', \mathcal{R}) \\
&= (\partial_{b,b_0}(\mathcal{A}_1[b_0/a] \times \mathcal{A}_2), \mathbf{r}', \mathbf{t}', \mathcal{R}) \qquad (1) \\
&= (\partial_{a,b}(\mathcal{A}_1 \times \mathcal{A}_2), \mathbf{r}', \mathbf{t}', \mathcal{R}) \qquad (2)
\end{aligned}
$$

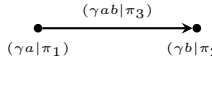
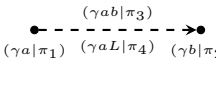
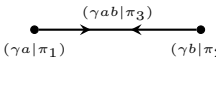
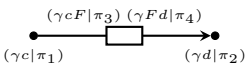
Synchronous Channels		
	$ab\|ab,\ \{(a,b,\gamma ab,\pi_3)\}$ 	$\begin{array}{\|c\|c\|}\hline & \mathbf{r}' \\ \hline a & (\gamma a,\pi_1) \\ b & (\gamma b,\pi_2) \\ \hline \end{array}$
	$ab\|ab,\ \{(a,b,\gamma ab,\pi_3)\}$ $a\bar{b}\|a,\ \{(a,\emptyset,\gamma aL,\pi_4)\}$ 	$\begin{array}{\|c\|c\|}\hline & \mathbf{r}' \\ \hline a & (\gamma a,\pi_1) \\ b & (\gamma b,\pi_2) \\ \hline \end{array}$
	$ab\|ab,\ \{(ab,\emptyset,\gamma ab,\pi_3)\}$ 	$\begin{array}{\|c\|c\|}\hline & \mathbf{r}' \\ \hline a & (\gamma a,\pi_1) \\ b & (\gamma b,\pi_2) \\ \hline \end{array}$
Asynchronous Channel		
	$c\|c,\ \{(c,\emptyset,\gamma cF,\pi_3)\}$  $d\|d,\ \{(\emptyset,d,\gamma Fd,\pi_4)\}$	$\begin{array}{\|c\|c\|}\hline & \mathbf{r}' \\ \hline c & (\gamma c,\pi_1) \\ d & (\gamma d,\pi_2) \\ \hline \end{array}$

Table 4.2: Stochastic Reo Automaton extended with reward information

where

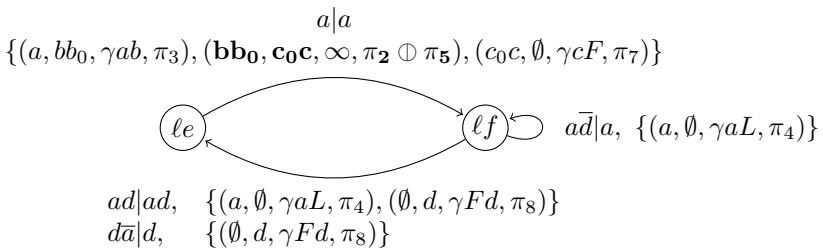
$$\begin{aligned}\mathbf{r}' &= \mathbf{r}'_1 \cup \mathbf{r}'_2 \cup \mathbf{r}'_{Sync(a_0b_0)} | (\Sigma_1 \cup \Sigma_2 \cup \{a_0, b_0\}) \setminus \{a, a_0, b, b_0\} \\ &= \mathbf{r}'_1 \cup \mathbf{r}'_2 | (\Sigma_1 \cup \Sigma_2) \setminus \{a, b\}\end{aligned}$$

since the $\mathbf{r}'$ function is defined only for boundary nodes, and

$$\begin{aligned}&\mathbf{t}'((q,p) \xrightarrow{gg'|ff'} (q',p')) = \mathbf{t}'_1(q \xrightarrow{g|f} q') \cup \mathbf{t}'_2(p \xrightarrow{g'|f'} p') \\ &\qquad\qquad \cup \{(\{a,a_0\},\{b_0,b\},\infty,\pi_a \odot \pi_b)\} \\ &\mathbf{t}'((q,p) \xrightarrow{gp^\sharp|f} (q',p)) = \mathbf{t}'_1(q \xrightarrow{g|f} q') \\ &\mathbf{t}'((q,p) \xrightarrow{gq^\sharp|f} (q,p')) = \mathbf{t}'_2(p \xrightarrow{g|f} p')\end{aligned}$$

Equality (1) follows by [19, Lemma 4.13] and (2) by an easily proven substitution property of node names.

The above implies that a reward sequence goes along with the relevant rate associated with a transition and does not affect the structure of Stochastic Reo Automata at all. Therefore, without loss of generality, Stochastic Reo Automata extended with reward information also support compositional specification and describe context-dependent connectors. Using the definitions for the composition of Stochastic Reo Automata in Section 4.2, the following figure shows the Stochastic Reo Automaton extended with reward information, corresponding to the modified LossyFIFO1 connector in Figure 4.8:



This result has the same structure as that of the Stochastic Reo Automaton in Figure 4.3, except for the 3-tuple $(bb_0, c_0c, \infty, \pi_2 \uplus \pi_5)$ which contains the reward information for the mixed node in the LossyFIFO1 connector in Figure 4.7.

4.4 Translation into CTMC

In this section, we show how to translate a Stochastic Reo Automaton into a homogeneous CTMC model. This translation is similar to the translation from QIA into CTMCs explained in Section 3.3, hence, to avoid repetition, in this section, we only show the different translation steps while using the same notations, if possible.

A CTMC model derived from a Stochastic Reo Automaton $(\mathcal{A}, \mathbf{r}, \mathbf{t})$ with $\mathcal{A} = (\Sigma, Q, \delta_{\mathcal{A}})$ is a pair (S, δ). With a set of boundary nodes $\Sigma' \subseteq \Sigma$, the set S_A and the preliminary set of request-arrival transitions of the CTMC derived for $(\mathcal{A}, \mathbf{r}, \mathbf{t})$ are defined as:

$$\begin{aligned} S_A &= \{\langle q, P\rangle \mid q \in Q,\ P \subseteq \Sigma'\} \\ \delta'_{Arr} &= \{\langle q, P\rangle \xrightarrow{\mathbf{r}(c)} \langle q, P \cup \{c\}\rangle \mid \langle q, P\rangle,\ \langle q, P \cup \{c\}\rangle \in S_A,\ c \notin P\} \end{aligned}$$

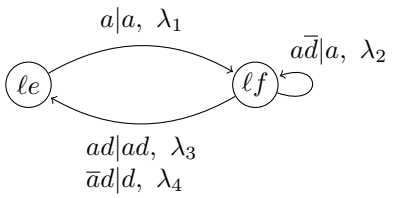
The set δ'_{Arr} is used to define δ_{Arr}, which includes preemptive request-arrivals arising in this translation process, used in the definition of δ above.

4.4.1 Synchronized data-flows

Synchronized data-flows are represented in a single transition of a Stochastic Reo Automaton. To divide this macro-step transition, corresponding to the synchronized data-flows, into a number of micro-step transitions, corresponding to each data-flow, the occurrence order of the synchronized data-flows need to be determined. This decision step is explained in Section 3.3.2 using a delay-sequence and Algorithm 3.3.1. Applying Algorithm 3.3.1 to the LossyFIFO1 example of Figure 4.3 yields the following result in Figure 4.9:

4.4.2 Deriving the CTMC

We now show how to derive the transitions in the CTMC model from the transitions in a Stochastic Reo Automaton. We do this in two steps:



λ_1 :	$(a, bc, \gamma ab)$; $(bc, \emptyset, \gamma cF)$
λ_2 :	$(a, \emptyset, \gamma aL)$
λ_3 :	$(a, \emptyset, \gamma aL)$ | $(\emptyset, d, \gamma Fd)$
λ_4 :	$(\emptyset, d, \gamma Fd)$

Figure 4.9: Extracting delay-sequences

1. For each transition $p \xrightarrow{g|f} q \in \delta_{\mathcal{A}}$, we derive transitions $\langle p, P\rangle \xrightarrow{\lambda} \langle q, P \setminus f\rangle$ for every set of pending requests P that suffices to activate the guard g (i.e., $\widehat{P} \leq g \setminus \widehat{\overline{\Sigma}}$), where λ is the delay-sequence associated with the set of 3-tuples $\mathbf{t}(p \xrightarrow{g|f} q)$. This set of derived transitions is defined below as δ_{Macro}.

2. We divide a transition in δ_{Macro} labeled by λ into a combination of micro-step transitions, each of which corresponds to a single event.

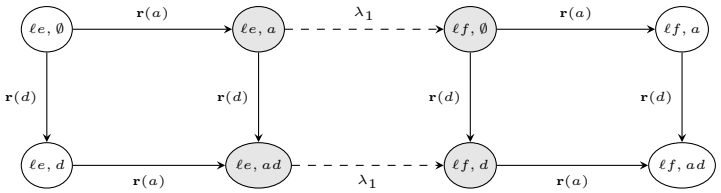
Given a Stochastic Reo Automaton $(\mathcal{A}, \mathbf{r}, \mathbf{t})$ with $\mathcal{A} = (\Sigma, Q, \delta_{\mathcal{A}})$ and a set of boundary nodes Σ', a macro-step transition relation for the synchronized data-flows is defined as:

$$\delta_{Macro} = \{\langle p, P\rangle \xrightarrow{\lambda} \langle q, P \setminus f\rangle \mid p \xrightarrow{g|f} q \in \delta_{\mathcal{A}},\ P \subseteq \Sigma',\ \widehat{P} \leq g \setminus \widehat{\overline{\Sigma}},\ \lambda = \mathbf{Ext}(\mathbf{t}(p \xrightarrow{g|f} q))\}$$

As an example of obtaining a macro-step transition relation, let us consider the transition $\ell e \xrightarrow{a|a,\lambda_1} \ell f$ with $\lambda_1 = (a, bc, \gamma ab)$; $(bc, \emptyset, \gamma cF)$ in Figure 4.9. Given the guard $g = a$ and the set of boundary nodes $\Sigma' = \{a, d\}$, $g \setminus \widehat{\overline{\Sigma}} = a \setminus \overline{a}\overline{b}\overline{c}\overline{d} = a$, and P is $\emptyset$, $\{a\}$, $\{d\}$, or $\{a, d\}$. Thus,

$$\widehat{P} = \begin{cases} a & \text{if } P = \{a\} \\ d & \text{if } P = \{d\} \\ ad & \text{if } P = \{a, d\} \\ \top & \text{otherwise} \end{cases}$$

Then, $\widehat{P} \leq g \setminus \widehat{\overline{\Sigma}}$ is satisfied when P is either $\{a\}$ or $\{a, d\}$, i.e., $a \leq a$ and $ad \leq a$. This generates the following macro-step transitions $\langle \ell e, a\rangle \xrightarrow{\lambda_1} \langle \ell f, \emptyset\rangle$ and $\langle \ell e, ad\rangle \xrightarrow{\lambda_1} \langle \ell f, d\rangle$, and these transitions are represented as dashed transitions in the state diagram that includes S_A and δ'_{Arr} as follows:



We explicate a macro-step transition with a number of micro-step transitions, each of which corresponds to a single data-flow. The detailed technical explanation on this division of a delay-sequence has been shown in Section 3.3.3. Thus, we skip the explanation of the division in this chapter.

The division into micro-step transitions ensures that each transition has a single 3-tuple in its label. Thus, the micro-step transitions can be extracted as:

$$\delta_{Proc} = \{\langle p, P\rangle \xrightarrow{v(\theta)} \langle p', P'\rangle \mid \langle p, P\rangle \xrightarrow{\theta} \langle p', P'\rangle \in div(t) \text{ for all } t \in \delta_{Macro}\}$$

As mentioned in Section 3.3.4, splitting synchronized data-flows allows non-interfering events, in particular request-arrivals, to interleave with micro-step events, disregarding the strict sense of the atomicity of the synchronized data-flows. The consideration of these preemptive request-arrivals is explained in Section 3.3.4.

4.4.3 Rewards

In this section, we show the translation from the Stochastic Reo Automata extended with reward information into CTMCs with state reward. As mentioned in Section 4.3, the reward sequence is independent of the structure of its Stochastic Reo Automaton. Thus, for the generation of CTMCs with state rewards, the translation from Stochastic Reo Automata into CTMCs can be reused with a small modification. When the CTMC (S, δ) is derived from a Stochastic Reo Automaton $(\mathcal{A}, \mathbf{r}, \mathbf{t})$ with $\mathcal{A} = (\Sigma, Q, \delta_{\mathcal{A}})$, the derived CTMC, thus, is $(S \times \mathbb{R}^*, \delta)$ for the extended Stochastic Reo Automaton $(\mathcal{A}, \mathbf{r}', \mathbf{t}', \mathcal{R})$ which is extended from $(\mathcal{A}, \mathbf{r}, \mathbf{t})$ with Q-algebra $\mathcal{R}$ for reward information:

- Stochastic Reo Automata
  - $\mathbf{r} : \Sigma \to \mathbb{R}^+$
  - $\mathbf{t} : \delta_{\mathcal{A}} \to 2^{\Theta}$ where $\Theta \subseteq 2^{\Sigma} \times 2^{\Sigma} \times \mathbb{R}^+$
- Stochastic Reo Automata with reward information
  - $\mathbf{r}' : \Sigma \to \mathbb{R}^+ \times \mathbb{R}^*$
  - $\mathbf{t}' : \delta_{\mathcal{A}} \to 2^{\Psi}$ where $\Psi \subseteq 2^{\Sigma} \times 2^{\Sigma} \times \mathbb{R}^+ \times \mathbb{R}^*$

Note that the extensions of $\mathbf{r}'$ and $\mathbf{t}'$ by adding a reward sequence do not affect the structure of a connector, the structural information of which is used for our translation. Thus, $\mathbf{r}$ and $\mathbf{t}$ in the previous translation method can be replaced with, respectively, $\mathbf{r}'$ and $\mathbf{t}'$ easily.

In general, a state has more than one outgoing transition, which illustrates more than one activity is possible in a state. These activities have generally different rewards. Thus, we need to calculate the proper state reward considering all possible rewards. For this purpose, state rewards of CTMCs are decided after the whole diagram is drawn. This requires that the reward sequences should be kept until the complete CTMC diagram is drawn. The following shows the translation method into CTMCs considering this requirement.

While the CTMC derived from a Stochastic Reo Automaton $(\mathcal{A}, \mathbf{r}, \mathbf{t})$ with $\mathcal{A} = (\Sigma, Q, \delta_{\mathcal{A}})$ is (S, δ), for the extended Stochastic Reo Automaton $(\mathcal{A}, \mathbf{r}', \mathbf{t}', \mathcal{R})$ with reward information, the complete CTMC diagram is described as a tuple (S, δ') where $S = S_A \cup S_M$ and $\delta' = \delta_{Arr} \cup \delta_{Proc} \subseteq S \times \mathbb{R}^+ \times \mathbb{R}^* \times S$. Each label on a transition in δ' is a pair of a rate, specifying request-arrivals and processing delay of the transition, and its relevant reward sequence.

For transitions of request-arrivals, given the Stochastic Reo Automaton $(\mathcal{A}, \mathbf{r}', \mathbf{t}', \mathcal{R})$ with $\mathcal{A} = (\Sigma, Q, \delta_{\mathcal{A}})$ and a set of boundary nodes $\Sigma' \subseteq \Sigma$, the set S_A and the preliminary set of request-arrival transitions of the CTMC are defined as:

$$\begin{aligned} S_A &= \{\langle q, P\rangle \mid q \in Q,\ P \subseteq \Sigma'\} \\ \delta'_{Arr} &= \{\langle q, P\rangle \xrightarrow{\mathbf{r}'(c)} \langle q, P \cup \{c\}\rangle \mid \langle q, P\rangle, \langle q, P \cup \{c\}\rangle \in S_A,\ c \notin P\} \end{aligned}$$

The set δ'_{Arr} is used to define δ_{Arr} below.

For the division of synchronized data-flows, a new delay-sequence is redefined with the 4-tuple $\psi \in \Psi$:

$$\psi ::= \epsilon \mid \mu \mid \psi|\psi \mid \psi;\psi$$

The characteristics of the new delay-sequence μ is inherited from the existing delay-sequence λ in Section 3.3.1, and μ can also be extracted by Algorithm 3.3.1. Thus, the division of synchronized data-flows is carried out by the method mentioned in Section 4.4.2. The arrangement of labels on the divided result is described as:

$$\delta_{Proc} = \{\langle p, P\rangle \xrightarrow{pair(\mu)} \langle p', P'\rangle \in div(t) \mid t \in \delta_{Macro}\}$$

For the preemptive request-arrivals, with a set of micro-step states S_M, obtained through the division of synchronized data-flows, the full set of request-arrival transitions is defined as:

$$\delta_{Arr} = \delta'_{Arr} \cup \{\langle p, P\rangle \xrightarrow{\mathbf{r}'(d)} \langle p, P \cup \{d\}\rangle \mid \langle p, P\rangle,\ \langle p, P \cup \{d\}\rangle \in S_M, d \in \Sigma,\ d \notin P\}$$

So far we have derived a complete CTMC diagram from a Stochastic Reo Automaton extended with reward information, whereby the calculation of state rewards is shown below.

A state reward is decided by the outgoing transitions from each state, since the real values in the sequence represent the amount of resources that are required or released (gained or lost) for a transition materializing a request-arrival or a data-flow. When a label on a transition in δ' is denoted by $\kappa \in K \subseteq \mathbb{R}^+ \times \mathbb{R}^*$, the state

reward is obtained by a function $reward : S \to \mathbb{R}^*$: for every transition $s \xrightarrow{\kappa_1} s_1 \in \delta'$(e.g., $\delta_{Arr} \cup \delta_{Proc}$)

$$reward(s) \quad = \quad \begin{cases} \dfrac{\bigoplus_{i=1}^{n}(rate(\kappa_i) \boxdot rew(\kappa_i))}{\sum_{i=1}^{n} rate(\kappa_i)} & \text{if } \exists\ s \xrightarrow{\kappa_2} s_2 \in \delta', \cdots, \\ & \exists\ s \xrightarrow{\kappa_n} s_n \in \delta' \text{ and } s_i \neq s_j \text{ if } i \neq j \\ rew(\kappa_1) & \text{otherwise} \end{cases}$$

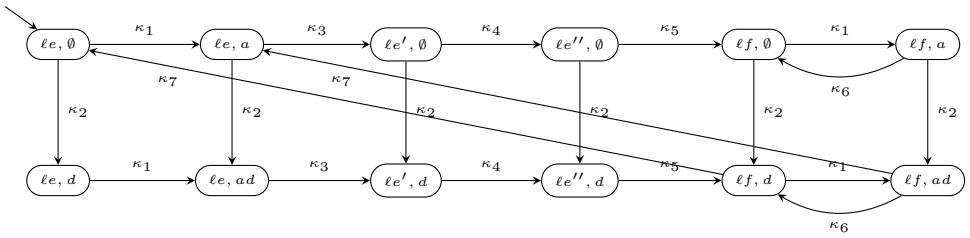
Note that $\boxdot : \mathbb{R}^+ \times \mathbb{R}^* \to \mathbb{R}^*$ is a function that returns the result of the multiplication of real number for its first parameter ($\mathbb{R}^+$) with every element in a reward sequence ($\mathbb{R}^*$), for instance, when $\pi = [\pi(0), \pi(1), \ldots, \pi(n)]$, the multiplication result of x and π is $x \boxdot \pi = [x \times \pi(0), x \times \pi(1), \ldots, x \times \pi(n)] \in \mathbb{R}^*$ where $\pi \in \mathbb{R}^*$ and $x, \pi(i) \in \mathbb{R}$ for $0 \leq i \leq n$. In addition, the summation $\boxplus : \mathbb{R}^* \times \mathbb{R}^* \to \mathbb{R}^*$ of reward sequences adds the values having the same index from each reward sequence. For example, for $\pi_1, \pi_2 \in \mathbb{R}^*,\ \pi_1 \boxplus \pi_2 = [\pi_1(1) + \pi_2(1), \pi_1(2) + \pi_2(2), \ldots, \pi_1(n) + \pi_2(n)] \in \mathbb{R}^*$. $\bigoplus_i^j$ is used to represent applying the summation $\boxplus$ for the reward sequences with the index from i to j.

After the calculation of state rewards, the extraction of the relevant rate for each transition δ' is done as:

$$\delta = \{s \xrightarrow{rate(\kappa)} s' \mid s \xrightarrow{\kappa} s' \in \delta'\}$$

The following example shows the calculation of state rewards from the complete CTMC diagram of the LossyFIFO1 connector extended with reward information.

Example 4.4.1. Consider the LossyFIFO1 example in Figure 4.7 and the CTMC diagram derived from it.



where the pairs κ_i of a rate and its corresponding reward sequence are shown below:

$$\begin{aligned} \kappa_1 &= (\gamma a, \pi_1) \\ \kappa_2 &= (\gamma d, \pi_6) \\ \kappa_3 &= (\gamma ab, \pi_3) \\ \kappa_4 &= (\infty, \pi_2 \oslash \pi_5) \\ \kappa_5 &= (\gamma cF, \pi_7) \\ \kappa_6 &= (\gamma aL, \pi_4) \\ \kappa_7 &= (\gamma Fd, \pi_8) \end{aligned}$$

Then, each state reward is given by:

$$
\begin{aligned}
reward_{\mathsf{LossyFIFO1}} = \{ \quad \langle \ell e, \emptyset \rangle &\mapsto \frac{(\gamma a \boxdot \pi_1) \boxplus (\gamma d \boxdot \pi_6)}{(\gamma a + \gamma d)} \\
\langle \ell e, a \rangle &\mapsto \frac{(\gamma ab \boxdot \pi_3) \boxplus (\gamma d \boxdot \pi_6)}{(\gamma ab + \gamma d)} \\
\langle \ell e', \emptyset \rangle &\mapsto \frac{(\infty \boxdot (\pi_2 \oplus \pi_5)) \boxplus (\gamma d \boxdot \pi_6)}{(\infty + \gamma d)} \\
\langle \ell e'', \emptyset \rangle &\mapsto \frac{(\gamma cF \boxdot \pi_7) \boxplus (\gamma d \boxdot \pi_6)}{(\gamma cF + \gamma d)} \\
\langle \ell f, \emptyset \rangle &\mapsto \frac{(\gamma a \boxdot \pi_1) \boxplus (\gamma d \boxdot \pi_6)}{(\gamma a + \gamma d)} \\
\langle \ell f, a \rangle &\mapsto \frac{(\gamma aL \boxdot \pi_4) \boxplus (\gamma d \boxdot \pi_6)}{(\gamma aL + \gamma d)} \\
\langle \ell e, d \rangle &\mapsto \pi_1 \\
\langle \ell e, ad \rangle &\mapsto \pi_3 \\
\langle \ell e', d \rangle &\mapsto \pi_2 \oplus \pi_5 \\
\langle \ell e'', d \rangle &\mapsto \pi_7 \\
\langle \ell f, d \rangle &\mapsto \frac{(\gamma a \boxdot \pi_1) \boxplus (\gamma Fd \boxdot \pi_8)}{(\gamma a + \gamma Fd)} \\
\langle \ell f, ad \rangle &\mapsto \frac{(\gamma aL \boxdot \pi_4) \boxplus (\gamma Fd \boxdot \pi_8)}{(\gamma aL + \gamma Fd)} \ \}
\end{aligned}
$$

In the case of $\langle \ell e', \emptyset \rangle$, its state reward is $[\infty, \ldots, \infty]/\infty$, i.e. the result value is not meaningful. However, the rate ∞ implies that its activity occurs immediately, and other possible activities can be ignored. Therefore, in this example, the state reward for the state $\langle \ell e', \emptyset \rangle$ is considered as $\pi_2 \oplus \pi_5$. ◊

4.5 Interactive Markov Chains and Reo

Interactive Markov Chains (IMCs) are a compositional stochastic model [43] which can be used to provide quantitative semantics to concurrent systems. In IMCs, delays can be represented by combinations of exponential delay transitions, it allows to accommodate non-exponential distributions within the models. That is, it can represent delays from the large class of phase-type distributions [72, 70] which can approximate general continuous distributions. This enables a more general usage of Stochastic Reo Automata, if IMCs are used instead of CTMCs as the translation target of Stochastic Reo Automata models.

In this section, we discuss to what extent IMCs are an appropriate semantic model for Stochastic Reo, instead of Stochastic Reo Automata. In addition, we provide a translation from Stochastic Reo into IMCs, which enables the use of the latter as an alternative target stochastic model.

4.5.1 Interactive Markov Chains

An IMC specifies a reactive system and is formally described as a tuple $(S, Act, \rightarrow, \Rightarrow, s_0)$ where S is a finite set of states; Act is a set of actions; s_0 is an initial state in S; $\rightarrow$ and $\Rightarrow$ are two types of transition relations:

- $\rightarrow \subseteq S \times Act \times S$ for *interactive* transitions and
- $\Rightarrow \subseteq S \times \mathbb{R}^+ \times S$ for *Markovian* transitions.

Thus, an IMC is a Labeled Transition System (LTS) if $\Rightarrow = \emptyset$ and $\rightarrow \neq \emptyset$, and is a CTMC if $\Rightarrow \neq \emptyset$ and $\rightarrow = \emptyset$.

Compared to other stochastic models such as CTMCs, the main strength of IMCs is their compositionality. Thus, one can generate a complex IMC as the composition of relevant simple IMCs, which enables compositional specification of complex systems.

Definition 4.5.1. ***(Product)*** *[43] Given two IMCs* $\mathcal{I}_1 = (S_1, Act_1, \rightarrow_1, \Rightarrow_1, s_{(1,0)})$ *and* $\mathcal{I}_2 = (S_2, Act_2, \rightarrow_2, \Rightarrow_2, s_{(2,0)})$*, the composition of* $\mathcal{I}_1$ *and* $\mathcal{I}_2$ *over a set of actions* A *is defined as* $\mathcal{I}_1 \times \mathcal{I}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, \Rightarrow, s_{(1,0)} \times s_{(2,0)})$ *where* $\rightarrow$ *and* $\Rightarrow$ *are defined as:*

$$
\begin{array}{rcl}
\rightarrow & = & \{(s_1, s_2) \xrightarrow{\alpha} (s_1', s_2') \mid \alpha \in A,\ s_1 \xrightarrow{\alpha}_1 s_1' \ \wedge\ s_2 \xrightarrow{\alpha}_2 s_2'\} \\
 & \cup & \{(s_1, s_2) \xrightarrow{\alpha} (s_1', s_2) \mid \alpha \notin A,\ s_2 \in S_2,\ s_1 \xrightarrow{\alpha}_1 s_1'\} \\
 & \cup & \{(s_1, s_2) \xrightarrow{\alpha} (s_1, s_2') \mid \alpha \notin A,\ s_1 \in S_1,\ s_2 \xrightarrow{\alpha}_2 s_2'\} \\
\Rightarrow & = & \{(s_1, s_2) \overset{\lambda}{\Longrightarrow} (s_1', s_2) \mid s_2 \in S_2\ ,\ s_1 \overset{\lambda}{\Longrightarrow}_1 s_1'\} \\
 & \cup & \{(s_1, s_2) \overset{\lambda}{\Longrightarrow} (s_1, s_2') \mid s_1 \in S_1\ ,\ s_2 \overset{\lambda}{\Longrightarrow}_2 s_2'\}
\end{array}
$$

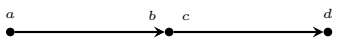
■

The product of interactive transitions is similar to the ordinary automata product, which includes interleaving and synchronized compositions of interactive transitions. The product of Markovian transitions consists of interleaved transitions only.

We now discuss IMCs from two different perspectives:

1. As a semantic model for Stochastic Reo: translating primitive Stochastic Reo channels into IMCs and then composing the derived IMCs using the product operation defined above; or
2. As an alternative translation target model: composing the Stochastic Reo Automata of primitive channels and then translating the composed Stochastic Reo Automaton into an IMC.

We show now that the first case is not adequate since it provides a wrong semantics for connectors that involve propagation of synchrony. For example, consider the following connector, denoted as 2sync, that consists of two Sync channels joined at nodes b and c.



The behavior of primitive channels consists of request-arrivals and data-flows which occur sequentially, i.e., data-flows follow request-arrivals. Both request-arrivals and data-flows are divided into two phases: an action and the random processing delay for each action. For instance, a request-arrival at node a consists of the arrival action at node a and waiting for the acceptance by node a. To reason about the end-to-end QoS, the IMCs for each Sync channel must have Markovian transitions for the random processing delays of both request-arrivals and data-flows. The two phases of channels must be considered sequentially, that is, the phase of random processing delays follows that of the action. Figure 4.10 shows the possible IMCs for the Sync channels ab and cd.

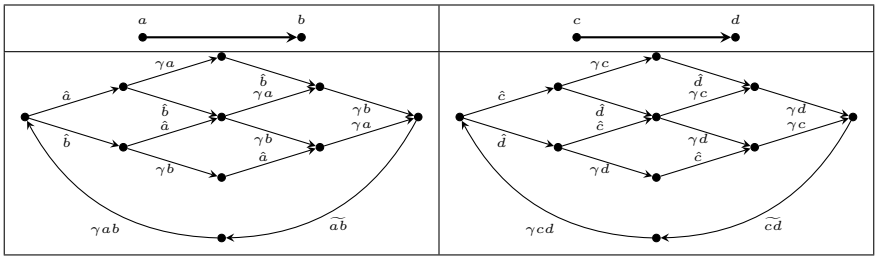


Figure 4.10: IMCs for each Sync channel

Here, we use '^' and '~' over node names in order to represent request-arrivals and data-flows, respectively. Rates for each request-arrival and each data-flow are represented with the prefix γ.

However, the composition of the IMCs for the two Sync channels does not capture the correct behavior of 2sync as specified by Reo. Figure 4.11 shows a fragment of the IMC product result where, for simplicity, the rest of the product result is omitted and represented as a cloud shape.

If we apply the assumption that the synchronization by joining nodes is an immediate action, then transitions with $(\hat{b}, \hat{c})$, γb, and γc labels are considered internal interactive transitions or discarded by certain refinements before or after the product. The result of the product and certain refinements is depicted in Figure 4.12.
Consider the diamond shape (1) in Figure 4.12, formed by the two data-flows from a to b (γab) and from c to d (γcd), which occur interleaved. In the 2sync circuit, these two data-flows occur sequentially, which means that data-flows do not occur concurrently. This example illustrates that using the concurrent composition of IMCs is not appropriate for specifying the behavior of connectors because it does not properly model the propagation of synchrony. It is natural and interesting to consider whether it is possible to adapt the composition operator of IMCs in order to delete unintended transitions (such as in the diamond shape (1) in Figure 4.12) and still retain a compositional model. However, we did not investigate this possibility since it is out of the scope of this thesis.

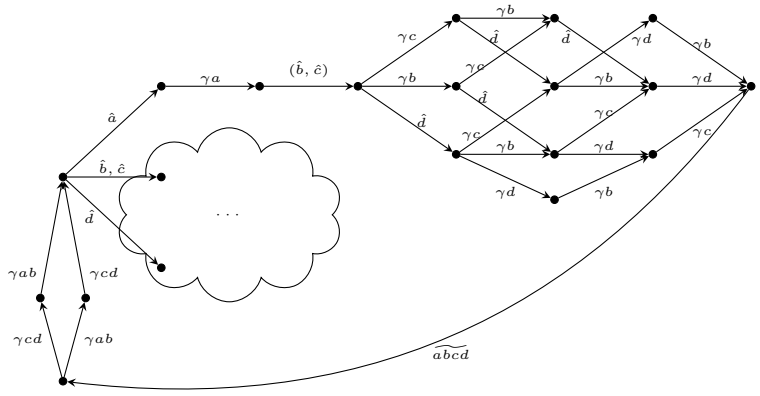


Figure 4.11: Composed IMC for 2sync

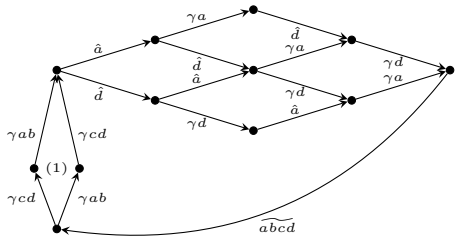


Figure 4.12: IMC after refinements on 2sync

We now show how IMCs can be used as a target stochastic model, instead of CTMCs. In this approach, the synchronization is considered in Stochastic Reo Automata, and we do not need to consider the IMC level refinements for synchronization such as the transitions with the labels $(\hat{b}, \hat{c})$, γb, and γc in Figure 4.11.

Since a Stochastic Reo Automaton does not have an initial state, the derived result is precisely an IMC transition system (IMCTS) [43], i.e., an IMC without an initial state. However, an initial state can be decided by the interpretation of the behavior of a connector. Thus, in this section, we consider the IMCTS derived from a Stochastic Reo Automaton as an IMC. An IMC derived from a Stochastic Reo Automaton $(\mathcal{A}, \mathbf{r}, \mathbf{t})$ with $\mathcal{A} = (\Sigma, Q, \delta_{\mathcal{A}})$ is a tuple $(S, Act, \rightarrow, \Rightarrow)$ where $S = S_A \cup S_M$ is the set of states. S_A represents the configurations of the system derived from its Stochastic Reo Automaton and the pending status of I/O requests; S_M is the set of states that represent the occurrences of synchronized data-flows and result from the micro-step divisions of the synchronized data-flows. In general, *Act* is a set of actions of the

arrival of a data item at a boundary node and synchronized data-flows through a connector. Thus, $Act = \Sigma' \cup Frs$ where Σ' is a set of boundary nodes, and Frs is a set of firings, i.e., the counterpart of f in a label on a transition $s \xrightarrow{g|f} s' \in \delta_{\mathcal{A}}$, $f \in Frs$. The relation $\rightarrow = \zeta_{Arr} \cup \zeta_{Proc} \subseteq (S \times 2^{\Sigma'} \times S) \cup (S \times 2^{Frs} \times S)$ is a set of interactive transitions, and $\Rightarrow = \delta_{Arr} \cup \delta_{Proc} \subseteq S \times \mathbb{R}^+ \times S$ is a set of Markovian transitions. The sets indexed with Arr and $Proc$ represent transitions for request-arrivals and data-flows, respectively.

A state in $S \subseteq Q \times 2^{\Sigma'} \times 2^{\Sigma'}$ represents three kinds of configurations: configurations of a connector (Q), the occurrences of actions (first $2^{\Sigma'}$), and the presence of the I/O requests pending on its boundary nodes (second $2^{\Sigma'}$), if any. The set of S_A and the preliminary sets of request-arrival transitions are defined as:

$$
\begin{aligned}
S_A &= \{(q, A, P) \mid q \in Q,\ P \subseteq A \subseteq \Sigma'\} \\
\zeta'_{Arr} &= \{(q, A, P) \xrightarrow{\hat{c}} (q, A \cup \{c\}, P) \mid (q, A, P),\ (q, A \cup \{c\}, P) \in \Sigma',\ c \notin A\} \\
\delta'_{Arr} &= \{(q, A, P) \xRightarrow{\mathbf{r}(c)} (q, A, P \cup \{c\}) \mid (q, A, P),\ (q, A, P \cup \{c\}) \in \Sigma',\ c \notin P\}
\end{aligned}
$$

The sets ζ'_{Arr} and δ'_{Arr} are used to define below ζ_{Arr} and δ_{Arr}, respectively.

As mentioned in Section 4.4.1, synchronized data-flows are described by a single transition in a Stochastic Reo Automaton. From the interactive transition perspective, the synchronized data-flows are also described by a single interactive transition. However, from the Markovian transition perspective in a continuous time domain, a transition corresponding to multiple synchronized data-flows needs to be divided into micro-step transitions. For this purpose, we reuse the delay-sequence which is extracted by Algorithm 3.3.1. We now derive transitions for synchronized data-flows in two steps:

1. For each transition $p \xrightarrow{g|f} q \in \delta_{\mathcal{A}}$, we derive interactive and macro Markovian transitions $(p, A, P) \xrightarrow{\widetilde{f}} (p, A \setminus f, P)$ and $(p, A \setminus f, P) \xrightarrow{\lambda} (q, A \setminus f, P \setminus f)$, respectively, for every set of pending requests P that suffices to activate the guard g ($\widehat{P} \leq g \setminus \widehat{\widehat{\Sigma}}$), where λ is the delay-sequence extracted by Algorithm 3.3.1 $\mathbf{Ext}(\mathbf{t}(p \xrightarrow{g|f} q))$. The sets of derived transitions are defined below as ζ_{Macro} and δ_{Macro} for interactive and macro Markovian transitions, respectively.

2. We divide a transition $s \xrightarrow{\lambda} s' \in \delta_{Macro}$ into a combination of micro-step transitions, each of which corresponds to a single event.

Given a Stochastic Reo Automaton $(\mathcal{A}, \mathbf{r}, \mathbf{t})$ with $=(Q, \Sigma, \delta_{\mathcal{A}})$ and a set of boundary nodes Σ', a macro-step transition for synchronized data-flows is defined as:

$$
\begin{aligned}
\zeta_{Macro} = \quad & \{(p, A, P) \xrightarrow{\widetilde{f}} (p, A \setminus f, P) \mid p \xrightarrow{g|f} q \in \delta_{\mathcal{A}},\ A \subseteq P \subseteq \Sigma',\ \widehat{P} \leq g \setminus \widehat{\widehat{\Sigma}}\} \\
\delta_{Macro} = \quad & \{(p, A, P) \xrightarrow{\lambda} (q, A, P \setminus f) \mid p \xrightarrow{g|f} q \in \delta_{\mathcal{A}},\ A \cap f = \emptyset,\ A \subset P \subseteq \Sigma', \\
& \qquad \widehat{P} \leq g \setminus \widehat{\widehat{\Sigma}},\ \lambda = \mathbf{Ext}(\mathbf{t}(p \xrightarrow{g|f} q))\}
\end{aligned}
$$

To derive an IMC from a Stochastic Reo Automaton, we reuse the function *nodes* and modify the definitions of functions *states* and *div* in Section 3.3.3. Then, $S_M =$

$state((p, A, P) \xrightarrow{\lambda} (q, A, P'))$ where

$$states((p, A, P) \xrightarrow{\lambda} (q, A, P')) = \begin{cases} \{(p, A, P), (q, A, P')\} & \text{if } \lambda = \theta \\ \bigcup states(m)\ \forall m \in div((p, A, P) \xrightarrow{\lambda} (q, A, P')) & \text{otherwise} \end{cases}$$

The function $div : \delta_{Macro} \to 2^{\delta_{Macro}}$ is defined as:

$$div((p, A, P) \xrightarrow{\lambda} (q, A, P')) = \begin{cases} \{(p, A, P) \xrightarrow{\theta} (q, A, P')\} & \text{if } \lambda = \theta \wedge \nexists (p, A, P) \xrightarrow{\theta} (p', A, P') \in \delta_{Macro} \\ div((p, A, P) \xrightarrow{\lambda_1} (p_{\lambda_1}, A, P'')) \cup div((p_{\lambda_1}, A, P'') \xrightarrow{\lambda_2} (q, A, P')) & \\ \qquad \text{if } \lambda = \lambda_1; \lambda_2 \text{ where } P'' = P \setminus nodes(\lambda_1) & \\ \{m_1 \boxtimes m_2 \mid m_i \in div((p, A, P) \xrightarrow{\lambda_i} (p_{\lambda_i}, A, P'')),\ i \in \{1, 2\}\} & \\ \qquad \text{if } \lambda = \lambda_1 | \lambda_2 \text{ where } P'' = P \setminus nodes(\lambda_i) & \\ \emptyset \qquad \text{otherwise} & \end{cases}$$

where the function $\boxtimes : \delta_{Macro} \times \delta_{Macro} \to 2^{\delta_{Macro}}$ computes all interleaving compositions of the two transitions as follows. For a transition $(p, A, P) \xrightarrow{\lambda_1|\lambda_2} (q, A, P') \in \delta_{Macro}$, $(p, A, P) \xrightarrow{\lambda_1} (p_{\lambda_1}, A, P \setminus nodes(\lambda_1))$ and $(p, A, P) \xrightarrow{\lambda_2} (p_{\lambda_2}, A, P \setminus nodes(\lambda_2))$ correspond to, respectively, m_1 and m_2 of the third condition in the definition of the div function. While m_1 and m_2 are handled by the div function recursively, some auxiliary states, i.e., $states(m_1)$ and $states(m_2)$, are generated. In the interleaving composition, m_1 can occur at any states that are generated by $states(m_2)$, and vice-versa. This interleaving composition of m_1 and m_2 is represented as:

$$\begin{aligned} m_1 \boxtimes m_2 = \ \{\ & div\big((p_1, A, P_1) \xrightarrow{\lambda_2} (p_{(1,\lambda_2)}, A, P_1 \setminus nodes(\lambda_2))\big), \\ & div\big((p_2, A, P_2) \xrightarrow{\lambda_1} (p_{(2,\lambda_1)}, A, P_2 \setminus nodes(\lambda_1))\big) \ \mid \\ & (p_1, A, P_1) \in states(m_1) \text{ and } (p_2, A, P_2) \in states(m_2)\ \} \end{aligned}$$

This composition is similar way to the function $\bowtie$ explained in Section 4.4.2. The only difference between these two functions is the structure of their states: CTMC states are elements of $Q \times 2^{\Sigma'}$, whereas IMC states are in $Q \times 2^{\Sigma'} \times 2^{\Sigma'}$ where Σ' is a set of boundary nodes in a Stochastic Reo connector.

The division into micro-step transitions ensures that each transition has a single 3-tuple in its label. Thus, the micro-step transitions can be extracted as:

$$\begin{aligned} \delta_{Proc} = \{(p, A, P) \xRightarrow{v(\theta)} (p', A, P') \mid & \\ (p, A, P) \xrightarrow{\theta} (p', A, P') \in div(t) \text{ for all } t \in \delta_{Macro}\} & \end{aligned}$$

As mentioned above, interactive transitions in ζ_{Macro} do not need to be divided, thus, $\zeta_{Proc} = \zeta_{Macro}$

Splitting synchronized data-flows allows non-interfering events to interleave with their micro-steps, disregarding the strict sense of their atomicity. In order to allow

such interleaving, we must explicitly add such request-arrivals. For a Stochastic Reo Automaton $(\mathcal{A}, \mathbf{r}, \mathbf{t})$ with $\mathcal{A} = (\Sigma, Q, \delta_{\mathcal{A}})$ and a set of micro-step states S_M, its full sets of request-arrival transitions, including its request-arrivals, are defined as:

$$\zeta_{Arr} = \zeta'_{Arr} \cup \{(p, A, P) \xrightarrow{\hat{d}} (p, A \cup \{d\}, P) \mid (p, A, P), (p, A \cup \{d\}, P) \in S_M,\ d \in \Sigma,\ d \notin A\}$$

$$\delta_{Arr} = \delta'_{Arr} \cup \{(p, A, P) \xrightarrow{\mathbf{r}(d)} (p, A, P \cup \{d\}) \mid (p, A, P), (p, A, P \cup \{d\}) \in S_M,\ d \in \Sigma,\ d \notin P\}$$

Applying this method, Figure 4.13 shows the IMC corresponding to our 2sync example. The derived result is similar to the IMC for a Sync in Figure 4.10 and captures the correct behavior of the 2sync connector. Note that we use an abbreviated notation for states, for example, we use p, ab, a instead of $(p, \{a, b\}, \{a\})$.

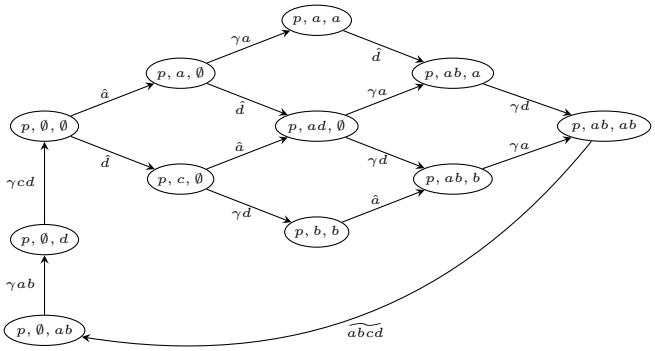


Figure 4.13: Derived IMC for 2sync

The foregoing illustrates that IMCs can serve as another alternative target model for the translation from Stochastic Reo Automata, instead of CTMCs. Although doing so does not exploit the compositionality of IMCs, translation into IMCs is still meaningful. The derived IMCs, for instance, can represent not only exponential distributions, but also non-exponential distributions, especially phase-type distributions. The analysis of IMCs is supported by tools such as the Construction and Analysis of Distributed Processes (CADP) [40]. CADP verifies the functional correctness of the specification of system behavior and also minimizes IMCs effectively [39]. Moreover, IMCs can be used in various other applications, such as Dynamic Fault Trees (DFTs) [21, 22, 23], Architectural Analysis and Design Language (AADL) [20, 25, 24], and so on [44].

4.6 Discussion

In this chapter, we introduced Stochastic Reo Automata by extending Reo Automata with functions that assign stochastic values of arrival rates and processing delay rates

to boundary nodes and channels in Stochastic Reo. This model is very compact and tractable to handle, compared to QIA. Various formal properties of Stochastic Reo Automata are obtained, reusing the formal justifications of the respective properties of Reo Automata [19], such as compositionality.

The technical core in this chapter shows the complexity of the original problem whence it stems from: derivation of stochastic models for formal analysis of end-to-end QoS properties of systems composed of services/components supplied by disparate providers, in their user environments. This complexity highlights the gross inadequacy of informal, or one-off techniques and emphasizes the importance of formal approaches and sound models that can serve as the basis for automated tools.

Stochastic Reo does not impose any restriction on the distribution of its annotated rates such as the rates for request-arrivals at channel ends or data-flows through channels. However, for the translation from Stochastic Reo into a homogeneous CTMC model, we considered only the exponential distributions for the rates. For more general usage of Stochastic Reo Automata, we also want to consider non-exponential distributions by considering phase-type distributions or using Semi-Markov Processes [50] as target models of our translation. A simulation engine [51, 89], already integrated into our toolset, the Extensible Coordination Tools (ECT) [35] environment, supports a wide variety of more general distributions for Stochastic Reo.

We discussed why IMCs are not an appropriate semantic model for Stochastic Reo, and showed the translation from Stochastic Reo into IMCs via Stochastic Reo Automata. A natural and interesting future work is to consider whether it is possible to adapt the composition operator of IMCs in order to delete unintended transitions that it currently produces in synchrony propagation scenarios, and still remain within a compositional framework.

Moreover, we have shown the extension of Stochastic Reo Automata with rewards information to specify stochastic behavior. Such an extension allows us to consider more general QoS aspects since, without rewards, Stochastic Reo Automata specify and reason about only timing information. The translation from the extended Stochastic Reo Automata into CTMCs with state reward is also shown.

So far, the translation from Stochastic Reo is carried out using QIA (instead of Stochastic Reo Automata) in our ECT environment. We are currently extending and improving these tools to use Stochastic Reo Automata, as well as the extension with reward information, so that the more compact sizes of the automata then allow us to analyze larger models.