# Stochastic models for quality of service of component connectors

Moon, Y.J.

# Chapter 3

# Quantitative Intentional Automata

## 3.1 Introduction

In Service-oriented Computing (SOC), services distributed over a network are composed according to the requirements of service consumers. Services are platform – and network – independent applications that support rapid, low-cost, loosely-coupled composition. Services run on the hardware of their own providers, in different containers, separated by firewalls and other ownership and trust barriers. Their composition requires additional mechanisms (e.g., process work-flow engines, connectors, and glue code) to impose some form of coordination (i.e., orchestration and/or choreography). Even if the quality of service (QoS) properties of every individual service and connector are known, it is far from trivial to build a model for and make statements about the end-to-end QoS of a composed system.

In CA, Reo Automata, and IA, mentioned in Chapter 2, the end-to-end QoS is not considered along with the specification of system behavior. In order to specify and reason about the end-to-end QoS of system behavior, Stochastic Reo was also introduced in Chapter 2. As the name reveals, Stochastic Reo is a stochastic extension of Reo and preserves the flexibility and the expressiveness of Reo for compositional construction of connectors. The aim of this chapter is to introduce a semantic model for Stochastic Reo.

This chapter consists of two parts. the first part introduces a semantic model for Stochastic Reo, *Quantitative Intentional Automata (QIA)* [6]. Actually, this semantic model is a stochastic extension of Intentional Automata (IA) [31]. We show the mapping between primitive Stochastic Reo channels and their corresponding QIA, as well as other operations such as the product.

The second part shows the translation from QIA into homogeneous CTMCs which are simple stochastic processes, widely used with efficient algorithms [85] for stochastic analysis.

## 3.2 Quantitative Intentional Automata

In this section, we introduce the notion of Quantitative Intentional Automata (QIA) which is designed as a stochastic extension of IA to provide an operational semantics for Stochastic Reo. Existing semantic models for Reo include IA[1] and CA. Whereas CA transitions describe system configuration changes, transitions of IA (and QIA) describe both the changes of system configuration as well as the changes of pending I/O requests. In CA, configurations are stored in the states, and processes causing state changes are shown in transition labels as a set of nodes where data are observed. Similarly, in IA (and QIA), the configurations of a system and the pending I/O requests are stored in the states. A data-flow or a firing through nodes causes changes in the system configuration, and arrivals of I/O requests at the nodes change the configurations of the pending I/O requests. These two different types of changes are distinguishably represented. (See Definition 3.2.1.)

**Definition 3.2.1 (Quantitative Intentional Automaton).** *A Quantitative Intentional Automaton is a tuple* $(Q, I, \Sigma, \rightarrow, \mathbf{r})$ *where*

- $Q \subseteq L \times 2^{\Sigma}$ *is a finite set of states, where*
  - $L$ *is a finite set of system configurations.*
  - $2^{\Sigma}$ *is a set of pending node sets, each element in* $2^{\Sigma}$ *describes the pending status in the current state.*
- $I \subseteq Q$ *is a set of initial states.*
- $\Sigma$ *is a finite set of nodes.*
- $\rightarrow \subseteq Q \times 2^{\Sigma} \times 2^{\Sigma} \times 2^{\Theta} \times Q$ *is the transition relation where* $\Theta \subseteq 2^{\Sigma} \times 2^{\Sigma} \times \mathbb{R}^{+}$ *such that for any* $I, O \subseteq \Sigma$ *and* $I \cap O = \emptyset$*, each* $(I, O, r) \in \Theta$ *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* $s \xrightarrow{R,F,D} s' \in \rightarrow$ *with* $D = \{(I_1, O_1, r_1), \ldots, (I_n, O_n, r_n)\}$, $F \setminus (I \cap O) = (I \cup O) \setminus (I \cap O)$ *where* $I = \bigcup_{1 \leq i \leq n} I_i$ *and* $O = \bigcup_{1 \leq i \leq n} O_i$.
- $\mathbf{r} : \Sigma \rightarrow \mathbb{R}^{+}$ *is a function that associates with each node its arrival rate.*

■

[1] IA is a general semantic model for component connectors. For a Reo connector, some invariants [31, Chapter 5] are required. Here and in the remainder of this chapter, IA are considered to be the extended version of IA, with the invariants, even though this will not be explicitly mentioned.

Note that for simplicity, here and throughout, we assume that all data constraints for transitions are *true* and thus abstract away the data constraints without loss of generality. In fact, the data constraints are used only for the nodes that fire, and they can be obtained from the transition with the same firing nodes in the corresponding CA of a QIA. In case of I/O request arrivals, there are no specific constraints on incoming data items, thus, the data constraints for I/O request arrivals are always *true*. In [31], for IA, a function $Q \to \mathcal{P}(2^{\Sigma} \times 2^{\Theta} \times Q)^{2^{\Sigma}}$ is used as an alternative.

In the QIA model for a Stochastic Reo connector, a transition $\langle q_1, R, F, D, q_2 \rangle$ is represented as $q_1 \xrightarrow{R,F,D} q_2$ where:

- $R$ is the set of nodes that interact with the environment of the nodes;
- $F$ is the set of nodes that fire and are released by the data-flows of the transition;
- $D \subseteq \Theta$ is the set of 3-tuples $\theta = (I, O, r)$ that correspond to individual data-flows in primitive Reo channels; the first two elements in $\theta$ depict the structural information of the relevant channel ends in the data-flow corresponding to $\theta$, e.g., input and output nodes for the data-flow; the last element shows the processing delay rate of the data-flow.

In case a transition corresponds to only request-arrivals, $D = \emptyset$. The arrival rates for I/O requests are given by the function $\mathbf{r}$. For instance, a QIA transition

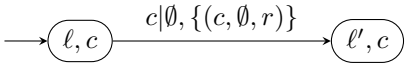
$$\langle \ell, \{a, b\} \rangle \xrightarrow{\emptyset,\{a,b\},\{(\{a\},\{b\},\gamma ab)\}} \langle \ell', \emptyset \rangle$$

encodes that a data-flow occurs from source node $a$ to sink node $b$ with the processing delay rate $\gamma ab$. Each element of a 3-tuple $\theta \in \Theta$ is accessed, respectively, by the projection functions $i : \Theta \to 2^{\Sigma}$, $o : \Theta \to 2^{\Sigma}$, and $v : \Theta \to \mathbb{R}^{+}$. Note that for readability, here and in the rest of this chapter, we use simplified representations for all sets used to describe QIA, such as $R$, $F$, and $D$ in a QIA transition. The curly brackets { and } for these sets are deleted, and the elements in the sets are arranged without commas. In addition, $R$ and $F$ are distinguished by a vertical bar '|', i.e., the above transition is represented as

$$\langle \ell, ab \rangle \xrightarrow{\emptyset|ab,\{(a,b,\gamma ab)\}} \langle \ell', \emptyset \rangle.$$

### 3.2.1 Invariants

Consider the following automaton, with two states.




This automaton satisfies Definition 3.2.1. However, it does not capture a behavior that corresponds to the semantics for a Reo connector. The reason is that node $c$ is

already pending (as indicated by the presence of $c$ in the configuration of the source state of the transition) and blocked to further request arrivals, therefore, it cannot interact with the environment as indicated by the new request arrival at node $c$ on the transition. The correct behavior of a Reo connector is subject to the same invariants mentioned for IA in Section 2.3.2. We recall these invariants. For an IA transition $\langle q, P\rangle \xrightarrow{R|F,D} \langle q', P'\rangle$, it is required that:

1. $F \subseteq R \cup P$
2. $R \subseteq F \cup P'$
3. $P \subseteq F \cup P'$
4. $P' \subseteq R \cup P$
5. $P \cap R = \emptyset$
6. $F \cap P' = \emptyset$

These invariants are also used for a QIA transition $\langle \ell, P\rangle \xrightarrow{R|F,D} \langle \ell', P'\rangle$, since in QIA, the function $\mathbf{r}$ and the set of 3-tuples $D$ in transition labels do not affect the structure on the transitions of QIA, which are decided by $F$.

The intuitive meaning of these invariants is explained in Section 2.3.2. Based on Definition 3.2.1 and these invariants, the appropriate QIA, as a semantic model for Stochastic Reo, corresponding to the primitive Stochastic Reo channels are presented in Figure 3.1. Note that here and the remainder of this chapter, for simplicity, when a set of 3-tuples is empty, i.e., transitions correspond to request-arrivals, we abstract it away. That is, only firing transitions include a set of relevant 3-tuples. The function $\mathbf{r}$ is shown as tables in Figure 3.1.

### 3.2.2 QIA composition

As mentioned in Section 2.2, a Stochastic Reo connector is obtained by composing primitive channels. Similarly, the QIA corresponding to a connector is also obtained by the composition of the QIA of its respective primitive channels. This composition is carried out in two operations:

1. **product** that plugs two automata together, considering synchronization and interleaving of the transitions from each automaton, and
2. **synchronization** that makes mixed nodes internal and filters firing transitions, taking into account the context-dependency of a Reo connector.

The QIA product is similar to the IA product in [31, Chapter 5]. However, this IA product does not account for the status of pending requests. In addition, we need to define how the QIA product handles the extended elements of the function $\mathbf{r}$ and the sets of 3-tuples in the transition labels of QIA. The function $\mathbf{r}$ associates arrival rates with the nodes in the set of nodes only, thus, it does not influence the structure of QIA. The set of 3-tuples in transition labels depends on firings. However, the converse is not true, i.e., the firing is not decided by the set of 3-tuples. In general, the product operation decides if firings are either composed together or interleave according to the synchronization constraints of the firings. Therefore, the function $\mathbf{r}$ and the set of 3-tuples do not affect the product definition, which as usual (e.g., CA and IA) primarily depends on firings. We define the QIA product considering synchrony of firings as follows:

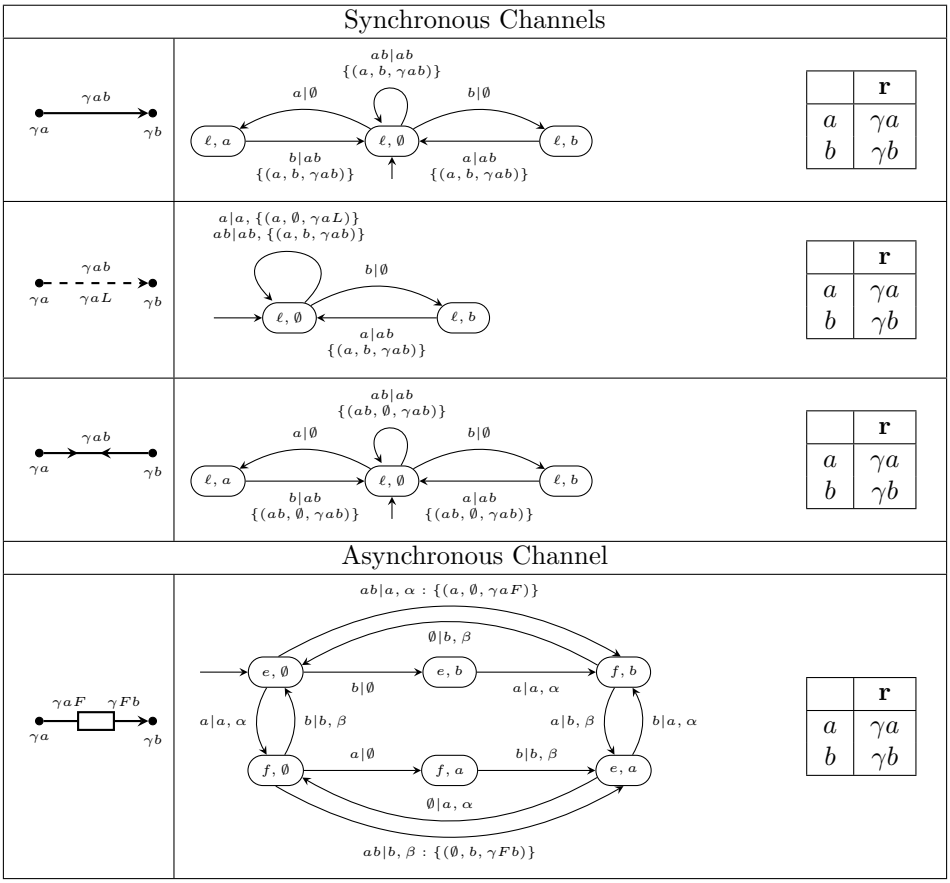



Figure 3.1: QIA for channels of Figure 2.3

**Definition 3.2.2 (Product of QIA).** *Given two QIA* $\mathcal{A} = (Q_1, I_1, \Sigma_1, \rightarrow_1, \mathbf{r}_1)$ *and* $\mathcal{B} = (Q_2, I_2, \Sigma_2, \rightarrow_2, \mathbf{r}_2)$*, their product is defined as* $\mathcal{A} \bowtie \mathcal{B} = (Q_1 \times Q_2, I_1 \times I_2, \Sigma_1 \cup \Sigma_2, \rightarrow, \mathbf{r}_1 \cup \mathbf{r}_2)$ *where* $\rightarrow$ *is given by following rules:*

1. $$\frac{\langle \ell_1, P_1\rangle \xrightarrow{R_1,F_1,D_1}_1 \langle \ell_1', P_1'\rangle \qquad F_1 \cap \Sigma_2 = \emptyset \qquad \forall \langle \ell_2, P_2\rangle \in Q_2 \text{ s.t. } P_2 \cap R_1 = \emptyset}{\langle (\ell_1, \ell_2), P_1 \cup P_2\rangle \xrightarrow{R_1,F_1,D_1} \langle (\ell_1', \ell_2), P_1' \cup P_2\rangle}$$

2. $$\frac{\langle \ell_2, P_2\rangle \xrightarrow{R_2,F_2,D_2}_2 \langle \ell_2', P_2'\rangle \qquad F_2 \cap \Sigma_1 = \emptyset \qquad \forall \langle \ell_1, P_1\rangle \in Q_1 \text{ s.t. } P_1 \cap R_2 = \emptyset}{\langle (\ell_1, l_2), P_1 \cup P_2\rangle \xrightarrow{R_2,F_2,D_2} \langle (\ell_1, \ell_2'), P_1 \cup P_2'\rangle}$$

3. $$\frac{\langle \ell_1, P_1\rangle \xrightarrow{R_1,F_1,D_1}_1 \langle \ell_1', P_1'\rangle \quad \langle \ell_2, P_2\rangle \xrightarrow{R_2,F_2,D_2}_2 \langle \ell_2', P_2'\rangle \quad F_1 \cap \Sigma_2 = F_2 \cap \Sigma_1 \ \wedge \ R_1 \cap P_2 = \emptyset = R_2 \cap P_1}{\langle (\ell_1, \ell_2), P_1 \cup P_2\rangle \xrightarrow{R_1 \cup R_2, F_1 \cup F_2, D_1 \cup D_2} \langle (\ell_1', \ell_2'), P_1' \cup P_2'\rangle}$$

■

Request-arrivals and data-flows are representative activities of Reo connectors. Request-arrivals are independent of synchrony or asynchrony of connector behavior, thus, the product result of transitions for request-arrivals interleave. On the other hand, data-flows are influenced by the synchrony or asynchrony of the behavior. Therefore, the product operation needs to consider the set of firing nodes, for example, $F_1 \cap \Sigma_2 = \emptyset$, $F_2 \cap \Sigma_1 = \emptyset$, and $F_1 \cap \Sigma_2 = F_2 \cap \Sigma_1$ in the definition. In addition, the consideration of request-arrivals ($R$) and existing pending status ($P$) is required to satisfy the aforementioned invariants. For example, consider the following two transitions:

1. $\langle \ell_1, a \rangle \xrightarrow{b|ab,D_1} \langle \ell_1', \emptyset \rangle$ with $\Sigma_1 = \{a, b\}$ 2. $\langle \ell_2, b \rangle \xrightarrow{c|bc,D_2} \langle \ell_2', \emptyset \rangle$ with $\Sigma_2 = \{b, c\}$

Taking into account only the synchrony, e.g, $\{a, b\} \cap \Sigma_2 = \{b, c\} \cap \Sigma_1$, the product result of transitions 1 and 2 is

$$\langle (\ell_1, \ell_2), ab \rangle \xrightarrow{bc|abc,D_1 \cup D_2} \langle (\ell_1', \ell_2'), \emptyset \rangle$$

However, this violates invariant 5 of $P \cap R = \emptyset$, i.e., $\{a, b\} \cap \{b, c\} = \{b\}$. Therefore, in the product operation, $P_1 \cap R_2 = \emptyset$, $P_2 \cap R_1 = \emptyset$, or $P_1 \cap R_2 = \emptyset = P_2 \cap R_1$ must be considered.

The product result of the set $D$ is the union of 3-tuple sets from each automaton. In order to keep QIA generally useful and compositional, and their product commutative, we avoid fixing the precise formal meaning of distributions of synchronized transitions composed in a product; instead, we represent the "processing delay rate" of their composite transition in the product automaton as the union of the processing delay rates of the synchronizing transitions of the two automata. How exactly these rates combine to yield the composite rate of the transition depends on the different properties of the distributions and their time ranges. For example, in the continuous-time case, no two events can occur at the same time; and the exponential distributions are not closed under taking maximum. In Section 3.3, we show how to translate a QIA to a CTMC using the union of the rates of the exponential distributions in the continuous-time case.

As an example, Figure 3.2 shows the product of a LossySync channel $ab$ and a Sync channel $bc$. For simplicity, we represent the set of 3-tuples in the labels of transitions only by their names and give them in the table below.

Note that the resulting automaton includes unintended transitions such as $\langle \ell, c \rangle \xrightarrow{a|a,\alpha} \langle \ell, c \rangle$ and $\langle \ell, bc \rangle \xrightarrow{a|a,\alpha} \langle \ell, bc \rangle$ which imply losing data items at node $a$ which violate context-dependency of the LossyFIFO1 connector. These unintended transitions are generated since the product operation does not consider mixed nodes as internal nodes that cannot interact with the outside. For this reason, we define a synchronization operation that makes mixed nodes internal and filters firing transitions taking into account the context-dependency of a Reo connector.

**Definition 3.2.3 (Synchronization).** *Given a QIA $\mathcal{A} = (Q, I, \Sigma, \rightarrow, \mathbf{r})$, synchronization of a mixed node $h \in \Sigma$, denoted by $synch[h](\mathcal{A})$, is equal to $(Q, I, \Sigma, \rightarrow', \mathbf{r}')$*

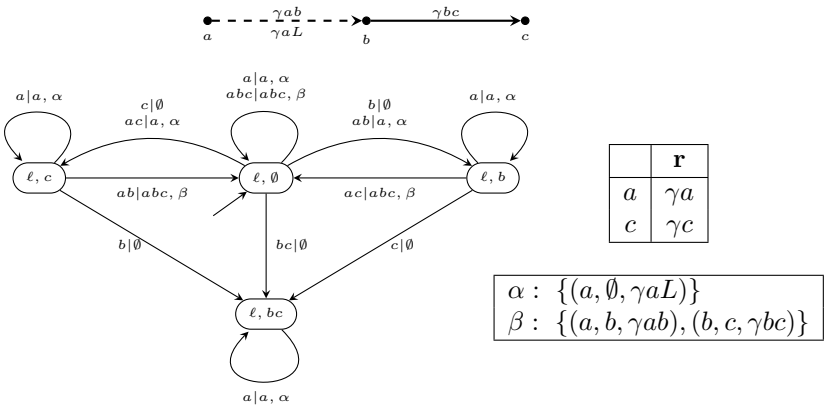



Figure 3.2: Product of a LossySync channel $ab$ and a Sync channel $bc$

*where*

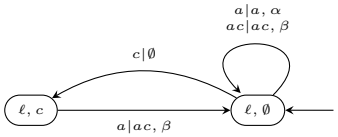
$$
\begin{array}{lcll}
\multicolumn{4}{l}{\mathbf{r}' \textit{ is } \mathbf{r} \textit{ restricted to the domain } \Sigma \setminus \{h\},\ r'(h) \textit{ is } \infty, \textit{ and}} \\
\rightarrow' & = & \{\langle \ell, P\rangle \xrightarrow{R,F,D} \langle \ell', P'\rangle \mid \langle \ell, P\rangle \xrightarrow{R \uplus \{h\}, F \uplus \{h\}, D} \langle \ell', P'\rangle\} & 1) \\
 & \cup & \{\langle \ell, P\rangle \xrightarrow{R,F,D} \langle \ell', P'\rangle \mid h \notin R \ \wedge\ h \notin F \ \wedge & \\
 & & \quad \nexists\ \langle \ell, P\rangle \xrightarrow{R',F',D'} \langle \ell'', P''\rangle \textit{ s.t. } R' = R \cup \{h\}\} & 2)
\end{array}
$$

*where* $\uplus : \Sigma \times \Sigma \to \Sigma$ *is a union restricted to disjoint sets.* ■

As an internal node, a mixed node does not interact with the environment and is always ready to dispense data-items, i.e., a mixed nodes deliver data items that it receives immediately. Thus, the synchronization operation restricts function $\mathbf{r}$ to only boundary nodes.

The product operation of QIA does not take into account context-dependency of connecting channels. The synchronization operation allows in an automaton (possibly resulting from the product of two automata) only firing transitions that respect the interaction with new environment. For example, consider the connector in Figure 3.2. In a LossySync channel $ab$, losing data at node $a$ occurs only when node $b$ is not pending. After the product with a Sync channel $bc$, node $b$ is always pending, and losing data occurs only when node $c$ is not pending. However, the state $\langle \ell, c\rangle$ in the product result has two firings for losing data at node $a$ and dispensing data from node $a$ to node $c$ via node $b$. The synchronization checks such situation and deletes unintended firings, i.e., it allows firings that 1) consider pending and firing at the mixed nodes as an immediate atomic activity or 2) are independent of mixed nodes. The QIA synchronization operation is analogous to the hiding operation [31, Chapter 4] for IA. The result of the synchronization operation on the product result

in Figure 3.2 is shown in Figure 3.3, which is the same as the original QIA for a LossySync channel.




$\alpha: \{(a, \emptyset, \gamma aL)\}$
$\beta: \{(a, b, \gamma ab), (b, c, \gamma bc)\}$

Figure 3.3: Synchronization result on QIA in Figure 3.2

The semantics of plugging two Reo connectors together on a common node $h$ is represented in QIA by first considering the product of their QIA and then applying the synchronization operation, i.e. $synch[h](\mathcal{A}_1 \bowtie \mathcal{A}_2)$. Thus, the product and the synchronization operations can be used to obtain, in a compositional way, the QIA of a connector built out of the primitive channels that comprise the connector. Given two QIA $\mathcal{A}_1$ and $\mathcal{A}_2$ with their node sets $\Sigma_1$ and $\Sigma_2$, respectively, sharing the common nodes $\Sigma_1 \cap \Sigma_2 = \{h_1, h_2, \ldots, h_k\}$, $synch[h_1](synch[h_2] \cdots (synch[h_k](\mathcal{A}_1 \bowtie \mathcal{A}_2)))$ represents the automaton corresponding to a connector. Note that the "plugging" order does not matter as synchronization interacts well with product.

**Example 3.2.4.** As a more complex example of the QIA composition, we apply the product and synchronization operations to the LossyFIFO1 example in Figure 2.5. The product result $\mathcal{A} = (Q, I, \Sigma, \rightarrow, \mathbf{r})$ of a LossySync channel $ab$ and a FIFO1 channel $bc$ is too big to draw and not readable. Instead of showing the whole figure of $\mathcal{A}$, we show $\rightarrow_{prod}$, the transitions that will be considered by the synchronization operation:

$$
\begin{aligned}
\rightarrow_{prod} = \{ \quad & \langle \ell e, \emptyset \rangle \xrightarrow{c|\emptyset} \langle \ell e, c \rangle, && \checkmark \\
& \langle \ell e, \emptyset \rangle \xrightarrow{\mathbf{ac}|\mathbf{a}} \langle \ell e, c \rangle, && \times \\
& \langle \ell e, \emptyset \rangle \xrightarrow{\mathbf{ac}\mathrm{b}|\mathbf{ab}} \langle \ell f, c \rangle, && \checkmark \\
& \langle \ell e, \emptyset \rangle \xrightarrow{\mathbf{a}|\mathbf{a}} \langle \ell e, \emptyset \rangle, && \times \\
& \langle \ell e, \emptyset \rangle \xrightarrow{\mathbf{ab}|\mathbf{ab}} \langle \ell f, \emptyset \rangle, && \checkmark \\
& \langle \ell f, \emptyset \rangle \xrightarrow{a|a} \langle \ell f, \emptyset \rangle, && \checkmark \\
& \langle \ell f, \emptyset \rangle \xrightarrow{c|c} \langle \ell e, \emptyset \rangle, && \checkmark \\
& \langle \ell f, \emptyset \rangle \xrightarrow{ac|ac} \langle \ell e, \emptyset \rangle, && \checkmark \\
& \langle \ell e, c \rangle \xrightarrow{\mathbf{a}|\mathbf{a}} \langle \ell e, c \rangle, && \times \\
& \langle \ell e, c \rangle \xrightarrow{\mathbf{ab}|\mathbf{ab}} \langle \ell f, c \rangle, && \checkmark \\
& \langle \ell f, c \rangle \xrightarrow{a|a} \langle \ell f, c \rangle, && \checkmark \\
& \langle \ell f, c \rangle \xrightarrow{\emptyset|c} \langle \ell e, \emptyset \rangle, && \checkmark \\
& \langle \ell f, c \rangle \xrightarrow{a|ac} \langle \ell e, \emptyset \rangle && \checkmark \quad \}
\end{aligned}
$$

Note that the system configuration $\ell e$ and $\ell f$ are the abbreviation of $(\ell, e)$ and $(\ell, f)$ where $\ell$ represents the system configuration of a LossySync channel $ab$, and $e$ and $f$ represent the configurations of, respectively, an empty and a full buffer of the FIFO1 channel $bc$. Formally, these system configurations must be written as $\langle(\ell, e), P\rangle$ where $P \subseteq \{a, b, c\}$, but for simplicity, we use the aforementioned abbreviations. The transitions with the bold labels represent the filtered transitions by the synchronization. The reason of this filtering is that the nodes in bold in these transitions are dependent on mixed nodes, thus, they must fire together with the mixed nodes. This dependency is shown by the presence of their counterparts that fire with the same nodes(represented in **bold**) as well as the mixed nodes (represented in roman, next to the nodes in bold). Each counterpart follows its relevant filtered transition above and belongs to the transition set 1) in Definition 3.2.3. The transitions with the black labels represent the independent firings of the mixed nodes and belong to the transition set 2) in Definition 3.2.3.

In this example, the three transitions $\langle \ell e, \emptyset\rangle \xrightarrow{a|a} \langle \ell e, \emptyset\rangle$, $\langle \ell e, \emptyset\rangle \xrightarrow{ac|a} \langle \ell e, c\rangle$, and $\langle \ell e, c\rangle \xrightarrow{a|a} \langle \ell e, c\rangle$, which have only the labels in bold, are deleted by the synchronization because they have counterparts that fire the pending mixed node $b$ since node $b$ is always ready to dispense data items as an internal node. The counterparts of these transitions are, respectively, $\langle \ell e, \emptyset\rangle \xrightarrow{ab|ab} \langle \ell f, \emptyset\rangle$, $\langle \ell e, \emptyset\rangle \xrightarrow{abc|ab} \langle \ell f, c\rangle$, and $\langle \ell e, c\rangle \xrightarrow{ab|ab} \langle \ell f, c\rangle$, firing mixed node $b$ is represented in red. The synchronization result of the product result $\mathcal{A}$ is shown in Figure 3.4.

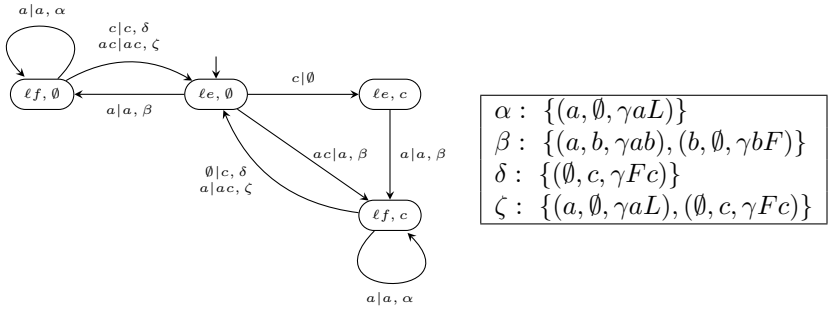



Figure 3.4: Corresponding QIA for LossyFIFO1 in Figure 2.5

◊

## 3.3 Translation into a stochastic model

In this section, we show how to translate QIA into a homogeneous CTMC model for stochastic analysis. In general, CTMCs are not compositional and large even for a

small system. That is, it is difficult to model CTMCs for complex systems directly. In our approach, modeling compositional behavior is carried out by QIA, and then corresponding CTMCs are derived from QIA which is considered as an intermediate model for this translation. Even though the resulting CTMCs are still big to handle, we can easily obtain CTMCs for complex systems via QIA. A homogeneous CTMC is a stochastic process with (1) discrete state space, (2) Markov property, (3) memoryless property, and (4) homogeneity in the continuous-time domain [43]. These properties yield efficient methodologies for numerical analysis. Note that here and in the remainder of this chapter, CTMCs are homogeneous even though it is not explicitly mentioned.

In the continuous-time domain, the exponential distribution is the only one that satisfies the memoryless property. Therefore, for the translation, we assume that the rates of request-arrivals and data-flows are exponentially distributed. However, note that such restriction did not appear in the QIA model. Other types of distributions can appear in the label of QIA, but then the target model has to be other than CTMCs.

A CTMC model derived from a QIA is a pair $(S, \delta)$ where $S = S_A \cup S_M$ is the set of states. $S_A$ represents the configurations of the system derived from its QIA including the pending status of I/O requests; $S_M$ is the set of states that result from the micro-step division of synchronized actions (see below). $\delta = \delta_{Arr} \cup \delta_{Proc} \subseteq S \times \mathbb{R}^+ \times S$, explained below, is the set of transitions, each labeled with a stochastic value specifying the arrival or the processing delay rate of the transition. $\delta_{Arr}$ and $\delta_{Proc}$ are defined in Section 3.3.4 and Section 3.3.3, respectively.

A state in QIA models a configuration of the connector, including the presence of the I/O requests pending on its boundary nodes, if any. Request-arrivals change system configuration only by changing the pending status of their respective boundary nodes. Data-flows corresponding to a transition in QIA change the system configurations, and release the pending I/O requests on their involved boundary nodes. In addition, data-flows depicted in a single transition illustrate multiple synchronized firings. In the following, we show how to deal with such request-arrivals and data-flows in an appropriate way for the translation from QIA into CTMCs.

In a CTMC model, the probability that two events (e.g., the arrival of an I/O request, the transfer of a data item, a processing step, etc.) happen at the same time is *zero*: only a single event occurs at a time. In compliance with this requirement, for a QIA $\mathcal{A} = (L \times 2^{\Sigma}, I, \Sigma, \rightarrow, \mathbf{r})$ and a set of boundary nodes $\Sigma'$, we define its set of request-arrival transitions, $\delta_{Arr}$, in several steps. The set $S_A$ and the preliminary set[2] of request-arrival transitions of the CTMC derived from $\mathcal{A}$ are defined as:

[2]In the process of generating CTMCs, some macro-step events (e.g., synchronized data-flows) are divided into several micro-step events. After that, independent events (e.g., request-arrivals) are considered as preemptive events between any two micro-step event. Before this division, we need to specify the transitions for respective synchronized data-flows. For this purpose, $S_A$ is obtained to describe source and target states of these transitions. The preliminary set of request-arrivals includes the transitions that connect the states in $S_A$, each of which corresponds to all possible request-arrivals at every connector configurations.

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

The set $\delta'_{Arr}$ is used in Section 3.3.4 to define the $\delta_{Arr}$ component of $\delta$.

As an example of obtaining $S_A$ and $\delta'_{Arr}$, recall the QIA for the LossyFIFO1 circuit in Figure 3.4. It has two system configuration of states $\ell e$ and $\ell f$, and its boundary nodes set $\Sigma'$ is $\{a, c\}$. Therefore:

$$\begin{aligned} S_A &= \{\ \langle \ell e, \emptyset\rangle,\ \langle \ell e, a\rangle,\ \langle \ell e, c\rangle,\ \langle \ell e, ac\rangle,\ \langle \ell f, \emptyset\rangle,\ \langle \ell f, a\rangle,\ \langle \ell f, c\rangle, \langle \ell f, ac\rangle\ \} \\ \delta'_{Arr} &= \{\ \langle \ell e, \emptyset\rangle \xrightarrow{\gamma a} \langle \ell e, a\rangle,\ \langle \ell e, \emptyset\rangle \xrightarrow{\gamma c} \langle \ell e, c\rangle,\ \langle \ell e, a\rangle \xrightarrow{\gamma c} \langle \ell e, ac\rangle, \\ &\quad \langle \ell e, c\rangle \xrightarrow{\gamma a} \langle \ell e, ac\rangle,\ \langle \ell f, \emptyset\rangle \xrightarrow{\gamma a} \langle \ell f, a\rangle,\ \langle \ell f, \emptyset\rangle \xrightarrow{\gamma c} \langle \ell f, c\rangle, \\ &\quad \langle \ell f, a\rangle \xrightarrow{\gamma c} \langle \ell f, ac\rangle,\ \langle \ell e, c\rangle \xrightarrow{\gamma a} \langle \ell f, ac\rangle\ \} \end{aligned}$$

The diagrams of $S_A$ and $\delta'_{Arr}$ are presented in Figure 3.5.

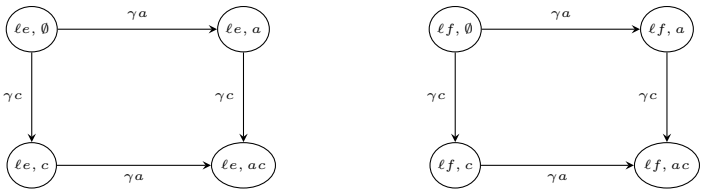



Figure 3.5: State diagram for request-arrivals

### 3.3.1 Micro-step transitions

The CTMC transitions associated with data-flows are more complicated because groups of synchronized data-flows are modeled as a single transition[3] in QIA, abstracting away their precise occurrence order. Therefore, we need to divide such synchronized data-flows into so-called micro-step transitions[4], respecting the connection information, i.e., the topology of a Reo connector, through which the data-flow occurs.

The connection information can be recovered from the 3-tuples in the label on each firing transition in a QIA, since the first and the second elements of a 3-tuple describe, respectively, the input and the output nodes involved in the data-flow of its transition, and the data-flow in the transition occurs from its input to its output nodes.

For example, the transition from state $\langle \ell e, \emptyset\rangle$ to state $\langle \ell f, \emptyset\rangle$ in the QIA of the LossyFIFO1 example in Figure 3.4 has $\{(a, b, \gamma ab), (b, \emptyset, \gamma bF)\}$ as its set of 3-tuples. The connection information inferred from this set states that the data-flows occur

[3]Note that here and in the remainder of this section, we skip to explicitly mention that a QIA transition $s \xrightarrow{R|F,D} s'$ satisfies $F \neq \emptyset\ \wedge\ D \neq \emptyset$ for its synchronized data-flows.

[4]This division delineates synchronized data-flows, not each data-flow itself.

from $a$ to the buffer through $b$. The transition is, thus, divided into two consecutive micro-step transitions $(a, b, \gamma ab)$ and $(b, \emptyset, \gamma bF)$.

Such data-flow information on each firing transition in a QIA is formalized by a *delay-sequence* defined by the following grammar:

$$\Lambda \ni \lambda ::= \epsilon \ \mid \ \theta \ \mid \ \lambda|\lambda \ \mid \ \lambda;\lambda$$

where $\epsilon$ is the empty sequence, and $\theta$ is a 3-tuple $(I, O, r)$ for a primitive Reo channel. $\lambda|\lambda$ denotes parallel composition, and $\lambda;\lambda$ denotes sequential composition. The empty sequence $\epsilon$ is an identity element for $|$ and ;, $|$ is commutative, associative, and idempotent, ; is associative and distributes over $|$. Most of properties of these compositional operators are intuitive, except for the distributivity of ;. The delay-sequence $\lambda$ extracted by the Algorithm 3.3.1 is in the format $\lambda = \lambda_1|\lambda_2|\ldots|\lambda_n$. Consider $\lambda = \lambda_1|\lambda_2 = (\theta_1;\theta_2)|(\theta_3;\theta_2)$[5]. Distributivity, that is, the property $(\theta_1;\theta_2)|(\theta_3;\theta_2) = (\theta_1|\theta_3);\theta_2$ is justified by the fact that $\theta_2$ is the delay of the same action and the other actions $\theta_1$ and $\theta_3$ in the composed delays $(\theta_1;\theta_2)$ and $(\theta_3;\theta_2)$ need to finish before the action corresponding to $\theta_2$ occurs. We use this distributivity law to generate compacter delay-sequences from the delay-sequences extracted in Section 3.3.2. For example, recall the delay-sequence $\lambda = (\theta_1|\theta_2)|(\theta_3;\theta_2)$. Then, $\lambda$ becomes $(\theta_1|\theta_3);\theta_2$ and it still preserves the sequential precedence of $\theta_1$ and $\theta_3$ over $\theta_2$ and shows the undetermined order between $\theta_1$ and $\theta_3$.

### 3.3.2 Extracting a delay-sequence

The delay-sequence corresponding to a set of 3-tuples associated with a transition in a QIA is obtained by Algorithm 3.3.1. Note that if the parameter of the function **Ext** is a singleton, then $\mathbf{Ext}(\{\theta\}) = \theta$ since $i(\theta) \cap o(\theta) = \emptyset$.

Intuitively, the **Ext** function delineates the set of activities that – at the level of a QIA – must happen synchronously/atomically, into its corresponding delay-sequences. If a certain data-flow associated with a 3-tuple $\theta_1$ explicitly precedes another one $\theta_2$, then $\theta_1$ is sequenced before $\theta_2$, i.e., encoded as $\theta_1;\theta_2$. Otherwise, they can occur in any order, encoded as $\theta_1|\theta_2$.

Applying Algorithm 3.3.1 to the LossyFIFO1 example in Figure 3.4 yields the following result shown in Figure 3.6, where the delineated results appear in the table.

The parameter $D$ of Algorithm 3.3.1 is a finite set of 3-tuples, and *Init*, *Post* and *toGo*, subsets of $D$, are also finite. Moreover, *Post* becomes eventually $\emptyset$ since *toGo* decreases during the procedure. Thus, we can conclude that Algorithm 3.3.1 always terminates.

A resulting delay-sequence $S$ extracted by Algorithm 3.3.1 is generated by the parallel composition of $\lambda_\theta$. The order of selecting $\theta$ from the set *Init* is not deterministic, thus, the resulting delay-sequence for the same input can be syntactically different,

[5]In general, the operators inside '()' have the highest order. Here and in the remainder of this thesis, we also follow this standard order without explicit mention.

**Algorithm 3.3.1**: Extraction of a delay-sequence out of a set $\Theta$ of 3-tuples

**Ext**$(D)$ where $D$ in $p \xrightarrow{R|F,D} q$
  $S = \epsilon$
  $toGo = D$
  $Init := \{\theta \in D \mid i(\theta) \cap o(\theta') = \emptyset \text{ for all } \theta' \in D\}$
  **for** $\theta \in Init$ **do**
    $\lambda_\theta := \theta$
    $Pre := \{\theta\}$
    $toGo := toGo \setminus Pre$
    $Post = \{\theta \in toGo \mid \exists \theta' \in Pre \ s.t. \ o(\theta') \cap i(\theta) \neq \emptyset\}$
    **while** $Post \neq \emptyset$ **do**
      $\lambda' := (\theta_1 | \cdots | \theta_k)$ where $Post = \{\theta_1, \cdots, \theta_k\}$
      $\lambda_\theta := \lambda_\theta; \lambda'$
      $Pre := Post$
      $toGo := toGo \setminus Pre$
      $Post := \{\theta \in toGo \mid \exists \theta' \in Pre \ s.t. \ o(\theta') \cap i(\theta) \neq \emptyset\}$
    **end while**
    $S := S|\lambda_\theta$
  **end for**
  **return** S

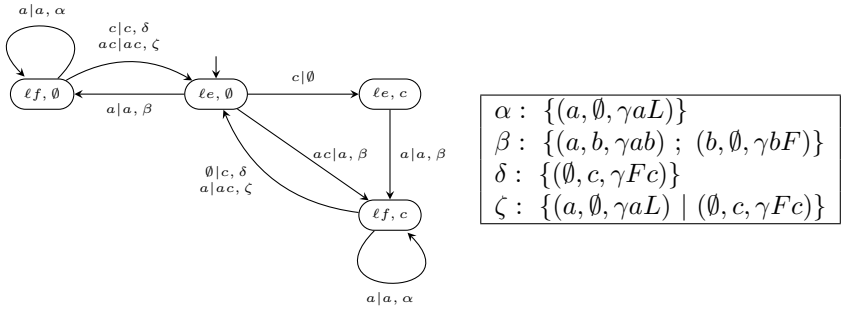



Figure 3.6: Applying Algorithm 3.3.1 to QIA in Figure 3.4

for example, $\lambda_\theta|\lambda_{\theta'}$ and $\lambda_{\theta'}|\lambda_\theta$ with $Init = \{\theta, \theta'\}$. However, the parallel composition operator | is commutative, thus, the composition order of | does not matter.

### 3.3.3 Dividing macro-step transitions with a delay-sequence

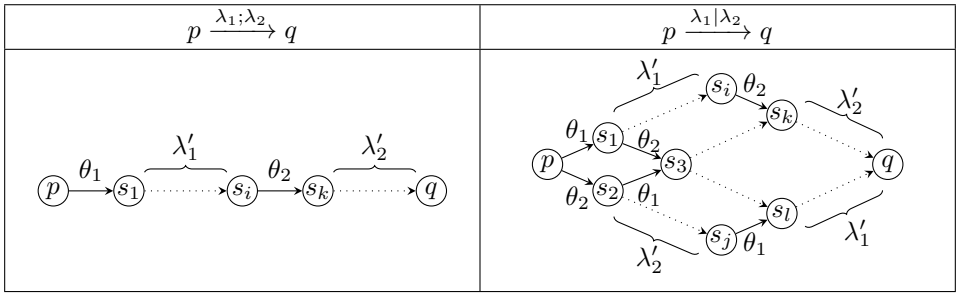
We now show how to derive the transitions in the CTMC model from the QIA transitions. In QIA, data-flows and request-arrivals can be put on a single transition,

whereas, in CTMCs, these two must be considered separately. For this purpose, we explore cases in which sole data-flows are possible. Recall the invariants of QIA in Section 3.2.1. A firing can occur when all its relevant nodes are pending. For a QIA transition $\langle q, P\rangle \xrightarrow{R|F,D} \langle q', P'\rangle$, the firing can occur if $F \subseteq P \cup R$. In compliance with this consideration, we derive the CTMC transitions in two steps:

1. For each QIA transition $\langle q, P\rangle \xrightarrow{R|F,D} \langle q', P'\rangle \in \rightarrow$ such that $F \neq \emptyset \ \wedge \ D \neq \emptyset$, we derive transitions $\langle q, P''\rangle \xrightarrow{\lambda} \langle q', P'' \setminus F\rangle$ where $P''$ is a node set that includes all the firing nodes of each QIA transition; $\lambda$ is the delay-sequence associated with the set of 3-tuples $D$ in the label on the transition. This set of derived transitions is defined below as $\delta_{Macro}$.

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

The following figure briefly illustrates the procedure mentioned above, for the two transitions $p \xrightarrow{\lambda_1;\lambda_2} q$ and $p \xrightarrow{\lambda_1|\lambda_2} q$ where $\lambda_1 = \theta_1; \lambda_1'$ and $\lambda_2 = \theta_2; \lambda_2'$:




A sequential delay-sequence $\lambda_1; \lambda_2$ allows for the events corresponding to $\lambda_1$ to occur before the ones corresponding to $\lambda_2$. For a parallel delay-sequence $\lambda_1|\lambda_2$, events corresponding to $\lambda_1$ and $\lambda_2$ occur interleaving each other, while they preserve their respective order of occurrence in $\lambda_1$ and $\lambda_2$. All indexed states $s_n$ are included in $S_M$ which consists of the states derived from the division of the synchronized data-flows into micro-step transitions. The formal description of dealing with these two delay-sequences is presented in the definition of a *div* function below, in which handling the respective delay-sequences correspond to the second and the third conditions of the *div* function.

Given a QIA $(Q, I, \Sigma, \rightarrow, \mathbf{r})$ and its boundary nodes set $\Sigma'$, a macro-step transition relation for the synchronized data-flows is defined as:

$$\delta_{Macro} = \{\langle p, P''\rangle \xrightarrow{\lambda} \langle q, P'' \setminus F\rangle \mid \langle p, P\rangle \xrightarrow{R|F,D} \langle q, P'\rangle \in \rightarrow,\ F \subseteq P'' \subseteq \Sigma',\ \lambda = \mathbf{Ext}(D)\}$$

As an example of obtaining a macro-step transition relation, consider the transition $\langle \ell e, \emptyset\rangle \xrightarrow{a|a,\beta} \langle \ell f, \emptyset\rangle$ with $\beta = (a, b, \gamma ab)\ ;\ (b, \emptyset, \gamma bF)$ in Figure 3.6. Given the firing

set $\{a\}$ and the boundary nodes set $\Sigma' = \{a, c\}$, $P''$ is $\{a\}$ or $\{a, c\}$, this generates the macro-step transitions $\langle \ell e, a\rangle \xrightarrow{\beta} \langle \ell f, \emptyset\rangle$ and $\langle \ell e, ac\rangle \xrightarrow{\beta} \langle \ell f, c\rangle$. Figure 3.7 shows a state diagram derived from the QIA of the LossyFIFO1 example in Figure 3.4 with the set of macro-step transitions $\delta_{Macro}$ and the preliminary set of request-arrival transitions $\delta'_{Arr}$, which are represented as dashed transitions.

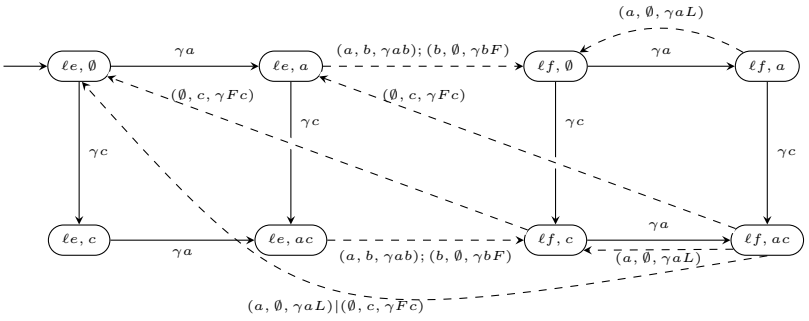



Figure 3.7: State diagram derived from the QIA in Figure 3.4 with $\delta_{Macro}$ and $\delta'_{Arr}$

We now explicate a macro-step transition with a number of micro-step transitions, each of which corresponds to a single data-flow. This refinement yields auxiliary states between the source and the target states of the macro-step transition. Let $\langle p, P\rangle$ be a source state for a data-flow corresponding to a 3-tuple $\theta$. The generated auxiliary states are defined as $\langle p_\theta, P \setminus nodes(\theta)\rangle$ where $p_\theta$ is just a label denoting that the data-flow corresponding to $\theta$ has occurred, and the function $nodes : \Lambda \to 2^\Sigma$ is defined for the delay-sequence level as:

$$nodes(\lambda) = \begin{cases} i(\theta) \cup o(\theta) & \text{if } \lambda = \theta \\ nodes(\lambda_1) \cup nodes(\lambda_2) & \text{if } \lambda = \lambda_1; \lambda_2 \ \vee \lambda = \lambda_1 | \lambda_2 \end{cases}$$

The set of such auxiliary states is obtained as $S_M = states(\langle p, P\rangle \xrightarrow{\lambda} \langle q, P'\rangle)$ where

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

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

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

where the function $\bowtie$: $\delta_{Macro} \times \delta_{Macro} \to 2^{\delta_{Macro}}$ computes all interleaving compositions of the two transitions as follows. For a transition $(p, R) \xrightarrow{\lambda_1|\lambda_2} (q, R') \in \delta_{Macro}$, $(p, R) \xrightarrow{\lambda_1} (p_{\lambda_1}, R \setminus nodes(\lambda_1))$ and $(p, R) \xrightarrow{\lambda_2} (p_{\lambda_2}, R \setminus nodes(\lambda_2))$ 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:

$$
\begin{aligned}
m_1 \bowtie m_2 = \ \{\ & div\big((p_1, R_1) \xrightarrow{\lambda_2} (p_{(1,\lambda_2)}, R \setminus nodes(\lambda_2))\big), \\
& div\big((p_2, R_2) \xrightarrow{\lambda_1} (p_{(2,\lambda_1)}, R \setminus nodes(\lambda_1))\big) \ \mid \\
& \qquad (p_1, R_1) \in states(m_1) \text{ and } (p_2, R_2) \in states(m_2)\ \}
\end{aligned}
$$

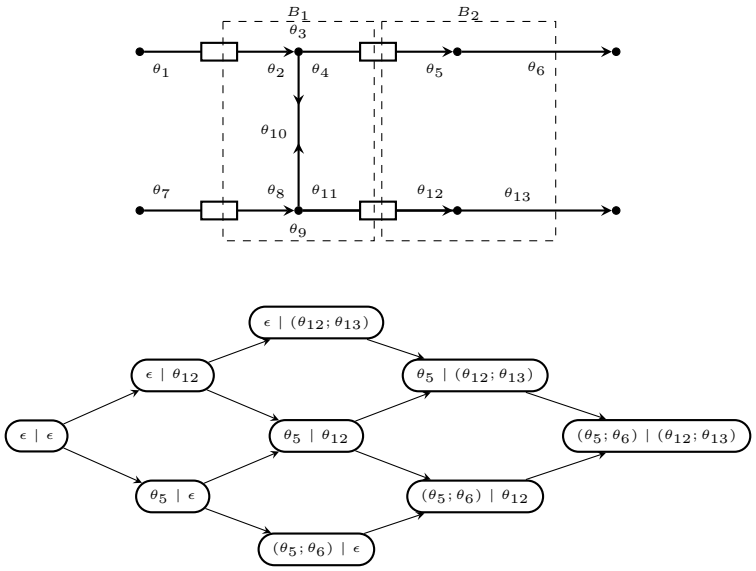
The following example shows the application of the function $div$ to a non-trivial delay-sequence, which contains a combination of sequential and parallel compositions.

**Example 3.3.1.** Consider the Stochastic Reo connector shown below. Every indexed $\theta$ is a rate for its respective processing activity, e.g., $\theta_2$ is the rate at which the top-left FIFO1 dispenses data through its sink end; $\theta_3$ is the rate at which the node replicates its incoming data, etc. Data-flows contained in boxed regions marked as $B_1$ and $B_2$ appear in $\delta_{Macro}$, derived from the QIA of this circuit, as two transitions with the delay-sequences of $\lambda_1$ and $\lambda_2$ where:

- from $B_1$: $\lambda_1 = ((\theta_2;\theta_3)|(\theta_8;\theta_9))\ ;\ (\theta_4|\theta_{10}|\theta_{11})$
- from $B_2$: $\lambda_2 = (\theta_5;\theta_6)\ |\ (\theta_{12};\theta_{13})$

To derive a CTMC, $\lambda_1$ and $\lambda_2$ must be divided into micro-step transitions. We exemplify a few of these divisions. For $\lambda_1$, the division of $(\theta_4|\theta_{10}|\theta_{11})$ is trivial since it contains only simple parallel composition. This division result is then appended to the division result of $(\theta_2;\theta_3)|(\theta_8;\theta_9)$, which has the same structure as that of $\lambda_2$. Thus, we show below the division result of $\lambda_2$ only.

In the following CTMC fragment, to depict which events have occurred up to a current state, the name of each state consists of the delays of all the events that have




occurred up to that state. The delay for a newly occurring event is appended at the end of its respective segment in the current state name.

This example shows that when a delay-sequence $\lambda$ is generated by parallel composition, the events in one of the sub-delay-sequences of $\lambda$ occur independently of the events in other sub-delay-sequences. Still events preserve their occurrence order within the sub-delay-sequence that they belong to. ◊

The division into micro-step transitions ensures that each such transition has a single 3-tuple in its label. As mentioned above, this 3-tuple includes the structural information and the processing delay rate of its relevant data-flow, but in CTMCs, only the processing delay rate is used. Thus, the extraction of processing delay rates from the micro-step transitions are defined as:

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

Figure 3.8 shows applying the division method and extracting rates from the LossyFIFO1 example in Figure 3.7. In Figure 3.8, the elements in $S_M$ appear in gray, and the micro-step transitions by the division method are represented as dashed transitions.

### 3.3.4 Preemptive request-arrivals

Synchronized data-flows in QIA are considered atomic, thus other events cannot interfere with them. However, splitting these data-flows allows non-interfering events to

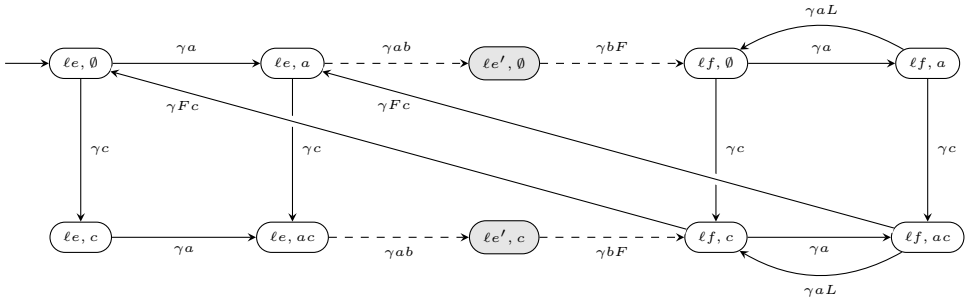



Figure 3.8: Division result of Figure 3.7

interleave with their micro-steps, disregarding the strict sense of their atomicity. For example, a certain boundary node unrelated to a group of synchronized data-flows can accept a data item between any two micro-steps. Since we want to allow such interleaving, we must explicitly add such request-arrivals. With a set of micro-step states $S_M$, its full set of request-arrival transitions, including its preliminary request-arrival set $\delta'_{Arr}$, is defined as:

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

Figure 3.9 shows the consideration of all possible preemptive request-arrivals for the division result in Figure 3.8, and the preemptive request-arrival is represented as a dashed transition. Thus, Figure 3.9 is the CTMC model $(S_A \cup S_M, \delta_{Arr} \cup \delta_{Proc})$ derived from the LossyFIFO1 example in Figure 2.5 via its QIA in Figure 3.4.

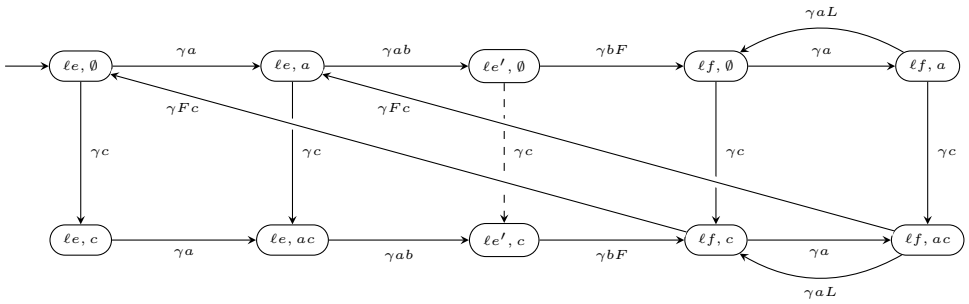



Figure 3.9: Derived CTMC of LossyFIFO1

## 3.4 Discussion

In this chapter, we introduced QIA as a semantic model for Stochastic Reo. This model specifies the behavior of a connector that coordinates services distributed over a network, along with its end-to-end QoS, specified as stochastic rates. QIA are an extension of IA, thus, QIA also consider both I/O request arrivals at channel ends and data-flows through channels separately. In contrast to IA, request arrivals and data-flows in QIA are considered stochastic activities. Considering the interaction with the environment of a connector, i.e., I/O request arrivals at channel ends, as a stochastic activity, QIA can specify and reason about end-to-end QoS aspects of system behavior. As IA capture the context-dependency of connectors, QIA, an extension of IA, also capture the context-dependency of connectors. As a complex connector is built out of the primitive Stochastic Reo channels, the QIA model corresponding to a complex connector is also obtained by composing the QIA models corresponding to the primitive Stochastic Reo channels that comprise the connector.

QIA are considered an intermediate model for translation into stochastic models, in particular CTMCs, for stochastic analysis. CTMCs are frequently used stochastic processes with some restrictions, such as discrete state space and Markov property. These restrictions (features) provide efficient algorithms for their numerical analysis [85]. For this purpose, we have shown the translation from Stochastic Reo into CTMCs in this chapter. Based on this method, a tool has been implemented in the Extensible Coordination Tools (ECT) [35], whose implementation details will be shown in Chapter 5. The CTMCs derived from Stochastic Reo via the QIA semantic model can be used for analysis of the stochastic behavior of Reo connectors.

In general, QIA are large models in terms of the number of states and transitions, because their configurations include not only data-flows, but also the interaction with the environment, in contrast to CA which consider the configurations of data-flows only. Thus, QIA quickly become too large to handle. Moreover, as a semantic model for Stochastic Reo, QIA must support the compositional semantics of a Stochastic Reo connector. However, the proof of the compositionality of QIA is far from trivial. Consequently, we designed a more compact and tractable semantic model, called Stochastic Reo Automata, which we present in Chapter 4.