# Encoding Context-Sensitivity in Reo into Non-Context-Sensitive Semantic Models (Technical Report)

Sung-Shik T.Q. Jongmans[1,⋆], Christian Krause[2,⋆⋆], Farhad Arbab[1]

[1] Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands
[2] Hasso Plattner Institute (HPI), University of Potsdam, Germany

**Abstract.** Reo is a coordination language which can be used to model the interactions among a set of components or services in a compositional manner using *connectors*. The language concepts of Reo include synchronization, mutual exclusion, data manipulation, memory and context-dependency. Context-dependency facilitates the precise specification of a connector's possible actions in situations where it would otherwise exhibit nondeterministic behavior. All existing formalizations of context-dependency in Reo are based on extended semantic models that provide constructs for modeling the presence and absence of I/O requests at the ports of a connector.
In this paper, we show that context-dependency in Reo can be encoded in basic semantic models, namely connector coloring with two colors and constraint automata, by introducing additional fictitious ports for Reo's primitives. Both of these models were considered as not expressive enough to handle context-dependency up to now. We demonstrate the usefulness of our approach by incorporating context-dependency into the constraint automata based Vereofy model checker.

⋆ Corresponding author. E-mail: jongmans@cwi.nl
⋆⋆ Supported by the research school in 'Service-Oriented Systems Engineering' at HPI

# Table of contents

# 1 Introduction

Over the past decades, *coordination languages* have emerged for the specification and implementation of interaction protocols for communicating software components. This class of languages includes Reo [Arb04], a platform for compositional construction of *connectors*. Connectors in Reo (or *circuits*) constitute the *glue* that holds component-based systems together. Moreover, once considered at a higher level of abstraction, connectors become components themselves; at this point in our presentation, we view connectors at this higher abstraction level. More specifically, we regard them as black boxes with a number of *ports* on which other components can issue *write requests* and *take requests*, collectively called *I/O-requests*, for *data items*. In the next section, we discuss connectors in more detail; here, we elaborate on I/O-requests.

Once issued on a port, an I/O-request remains *pending* until it *succeeds*. In case of a successful write request, the component that issued the request passes a data item through this port to the connector. Conversely, a successful take request causes the connector to pass a data item to the component. What happens to a data item after a component dispatches it to a connector depends on the specification of this connector. For instance, the connector might route the data item to another port, store it for some time in a buffer, or boldly send it into oblivion. Based on how connectors route data items, we can distinguish several behavioral properties, including *context-sensitivity* or *context-dependency*, the topic of this report.

Informally, the behavior of context-sensitive connectors does not depend only on their own current state, but also on the presence or absence of I/O-requests on their ports—their context. In contrast, the behavior of context-*in*sensitive connectors depends only on their own current state. To illustrate the concept of context-sensitivity, we consider the LossySync connector, which coordinates the interaction between two components: a *writer* and a *taker*. Suppose the writer issues as write request. If the taker is prepared to receive data—i.e., it has issued a take request—LossySync properly relays a data item from the writer to the taker. If the taker, however, has not issued a take request, LossySync still accepts a data item of the writer—i.e., the write request succeeds—but loses it without ever passing it to the taker. Since the behavior of LossySync depends on the presence or absence of take requests, it exhibits context-dependent behavior.

Several formal models for describing the behavior of Reo connectors exist, but not all of them have constructs for context-dependency. For example, the early models (e.g., *constraint automata* [BSAR06]), although attractive because of their simplicity, lack such constructs. These models implement context-sensitivity as non-determinism. In an attempt to mend this deficiency, more recent models incorporate constructs for context-dependency, but at the cost of more complex formalisms (e.g., the *3-coloring model* [CCA07]). As a result, the algorithms for their simulation and verification suffer from a high computational complexity, which makes these models less attractive in practice.

*Contributions* We show that some of the simple semantic models *can* describe the behavior of context-dependent connectors, namely the *2-coloring model* [CCA07] and constraint automata. More specifically, we define an operator that transforms a connector with 3-coloring semantics to one with 2-coloring semantics, while preserving its context-sensitive behavior. In a similar spirit, we define an operator that transforms a connector with 2-coloring semantics to one with constraint automaton semantics. To illustrate its merits, we show how our approach enables the verification of context-dependent connectors with the Vereofy model checker (considered impossible up to now). Other applications include context-sensitive *connector decomposition* [PSAB11], and, as we speculate, a more efficient implementation of Reo.

*Outline* In Section 2 (page 2), we give an overview of connectors in Reo. We discuss their structure and two formalisms for describing their behavior: coloring models and constraint automata. In Section 3 (page 15), we show how to transform context-dependent connectors with 3-coloring semantics to corresponding connectors with 2-coloring semantics. We introduce a transformation operator, prove its correctness, show its distributivity properties, and define its inverse. In Section 4 (page 44), we show how to transform connectors with 2-coloring semantics to corresponding connectors with constraint automaton semantics. The structure of this section resembles the structure of the section preceding it: we introduce a transformation operator, prove its correctness, and show its distributivity properties. In addition, we discuss an application of the two transformation operators combined: model checking with Vereofy. In Section 5 (page 55), we discuss related work. Section 6 (page 57) concludes this report.

# 2 Connectors in Reo

In this section, we discuss the structure of Reo connectors and two formalisms for describing their behavior: coloring models and constraint automata. A more comprehensive overview of Reo appears in [Arb04]. Before diving into the details, we make a remark on terminology: henceforth, we write "connector" or "circuit" to refer to both the structure and the *intended* behavior of a communication medium between software components. Note that the intended behavior of a connector and its behavior as described by some semantic model do not always coincide (e.g., the ordinary constraint automata of the LossySync connector).

In our presentation, we decouple the structure of a connector from its behavior and discuss both concepts individually: we treat connector structures in Section 2.1 (page 2) and the behavior of connectors in Section 2.2 (page 5). This separation allows us to associate different behavioral models with the same connector structure in the subsequent sections.

## 2.1 Structure

In the introduction, we already hinted at some structural properties of connectors: they have ports at which components can issue I/O-requests. Here, we

proceed by generalizing the concept of ports to that of *nodes*: entities through which data items can *flow* and at which *at most* two components or connectors can issue I/O-requests.[3] A connector then consists of a set of nodes, and its ports or *boundary nodes* constitute a special subset: they accept I/O-requests from the outside world. In addition to boundary nodes, connectors can have *internal nodes*. Contrary to the former, internal nodes cannot accept I/O-requests from the outside world, but play an important role in the relaying of data items through the connector by routing them past *primitives*. Primitives connect nodes to each other and serve as the elementary communication mediums through which data items can flow. If a data item flows from one node to another, we say that these nodes *fire*. Formally, we define a primitive as a list of (indexed) nodes, and we specify for each such node whether it accepts write requests—i.e., an *input node*—or take requests—i.e., an *output node*.

**Definition 1 (Universe of nodes [Cos10]).** $\mathcal{N}\mathsf{ode}$ *is the set of nodes.*

**Definition 2 (Primitive [Cos10]).** *Let* $N \subseteq \mathcal{N}\mathsf{ode}$*. A primitive* $e$ *over* $N$ *with arity* $k$ *is a list* $(n_1^{j_1}, \ldots, n_k^{j_k})$ *such that, for all* $1 \leq i, i' \leq k$, $N = \{\, n_l \mid 1 \leq l \leq k \,\}$ (NODES)*,* $j_i \in \{\, \mathrm{i}, \mathrm{o} \,\}$ (IO)*, and* $[i \neq i'$ *iff* $n_i \neq n_{i'}]$ (UNIQUENESS)*.*

Importantly, we do *not* associate primitives with any behavior; we use them merely as a convenient construct for describing the structure of elementary mediums (specifically, the *direction* in which data items flow through a connector). To describe what happens when a data item flows through a primitive $e$, we lift $e$ to a *primitive connector*: a connector without internal nodes and consisting of a single primitive, namely $e$, that connects all the (boundary) nodes.[4] Shortly, to illustrate this, we define some of the common primitives in terms of (the structure of) their corresponding primitive connectors. First, however, we give the formal definition of connector structures.

**Definition 3 (Connector structure [Cos10]).** *Let* $N \subseteq \mathcal{N}\mathsf{ode}$*. A connector structure* $C$ *over* $N$ *is a tuple* $\langle B, E \rangle$ *such that* $E$ *is a set of primitives* (PRIMS)*,* $N = \bigcup_{e \in E} \{\, n \in N' \mid e$ *is a primitive over* $N' \,\}$ (NODES)*, and* $\emptyset \neq B \subseteq N$ (B-NODES)*.*

To illustrate the definition of connector structures, a graphical representation of some of the common primitive connectors appears in Figure 1 (page 4); their formal definitions occur below. We make the structural equality between Sync, LossySync, and FIFO explicit by combining their definitions, but remark that

[3] In [Arb04], Arbab does not impose the restriction that at most two components or connectors can issue I/O-requests on a node: in his presentation, any number of components or connectors may do this. For simplicity, however, we choose to follow [CCA07,Cos10], which impose the same restriction as we do, without loss of generality: to model the merger/replicator-semantics of Arbab's nodes, we use explicit Merger and Replicator primitives.

[4] In [Cos10], primitives *do* have behavior. Here, we define the semantics of primitives only in terms of primitive connectors for the sake of simplicity of the presentation in Section 3 (page 15) and Section 4 (page 44).

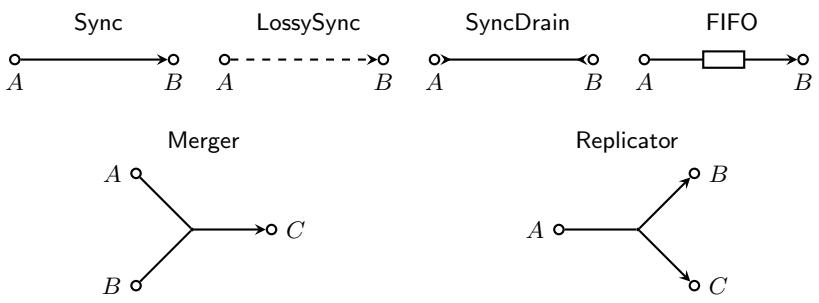


**Fig. 1.** Pictorial representation of common primitives.

they behave differently (as suggested also by the differences between their pictorial representations). We defer this topic for now, however, and revisit it in Section 2.2 (page 5).

$$\begin{aligned}
\mathsf{Sync}, \mathsf{LossySync}, \mathsf{FIFO} &\triangleq \langle \{ A, B \}, \{ (A^{\mathrm{i}}, B^{\mathrm{o}}) \} \rangle \\
\mathsf{SyncDrain} &\triangleq \langle \{ A, B \}, \{ (A^{\mathrm{i}}, B^{\mathrm{i}}) \} \rangle \\
\mathsf{Merger} &\triangleq \langle \{ A, B, C \}, \{ (A^{\mathrm{i}}, B^{\mathrm{i}}, C^{\mathrm{o}}) \} \rangle \\
\mathsf{Replicator} &\triangleq \langle \{ A, B, C \}, \{ (A^{\mathrm{i}}, B^{\mathrm{o}}, C^{\mathrm{o}}) \} \rangle
\end{aligned}$$

The primitive connectors that we introduced above cover only very simple communication protocols. To model and implement more sophisticated schemes of interaction, we can construct complex connectors from simpler constituents by means of *composition*. The following definition describes this concept formally in terms of a composition operator for connector structures.

**Definition 4 (Composition of connector structures).** *Let* $C_1 = \langle B_1, E_1 \rangle$ *and* $C_2 = \langle B_2, E_2 \rangle$ *be connector structures. Their composition, denoted* $C_1 \boxtimes C_2$*, is a connector structure over* $N_1 \cup N_2$ *defined as:*

$$C_1 \boxtimes C_2 = \langle (B_1 \cup B_2) \setminus (B_1 \cap B_2), E_1 \cup E_2 \rangle$$

Thus, to compose two connectors, we merge their sets of nodes, compute a new set of boundary nodes, and merge the sets of primitives that constitute them. A pictorial representation of three composed connectors appears in Figure 2 (page 5). We depict boundary nodes by white circles and internal nodes by black circles. Furthermore, for simplicity, we collapse consecutive instances of Merger and Replicator into single (black) circles in the depiction of ExclRouter and SyncFIFO, and we replace the two occurrences of ExclRouter by diamonds in the depiction of SyncFIFO. The latter does not cause ambiguity, because both diamonds have exactly one incoming arrow (implicitly attached to the input node of ExclRouter) and two outgoing arrows (implicitly attached to the two output nodes of ExclRouter).

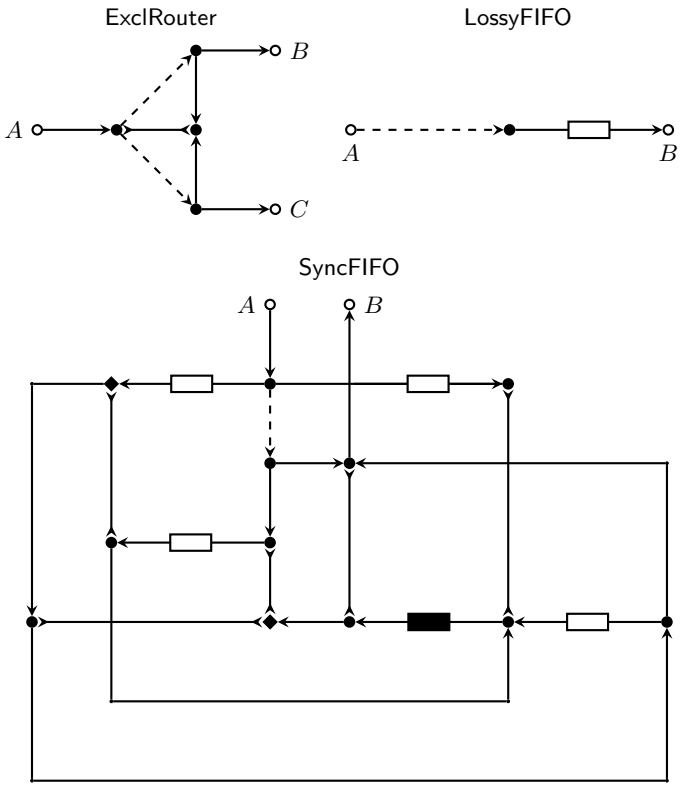


**Fig. 2.** Pictorial representation of ExclRouter, LossyFIFO and SyncFIFO.

## 2.2 Behavior

We proceed with two formalisms for describing the behavior of connectors. Before we discuss these—i.e., coloring models and constraint automata—we sketch the behavior of the six primitive connectors and the three composed connectors that we presented in the previous subsection; these connectors reoccur in the sequel.

*Primitive connectors* Sync routes a data item from its input node to its output node only if it has pending write and take requests on both of its nodes.

LossySync behaves almost indistinguishably from Sync, but as described in the introduction, loses a data item if its input node has a pending write request, while its output node has no pending take request.

SyncDrain accepts data items from its two input nodes only if they both have a pending write request. All the data items passed to SyncDrain disappear: they never reach another component or connector.

In contrast to the previous three connectors, FIFO has a *buffer* to store data items in. This buffer causes FIFO to exhibit different states, EMPTY and FULL, which influence its behavior. In the EMPTY state, a write request on the input

node of FIFO causes a data item to flow into the buffer, while a take request on its output node remains pending. Conversely, in the FULL state, a write request on its input node remains pending, while a take request on its output node causes a data item to flow from the buffer to the output node. We call FIFO a *state-dependent* or *state-sensitive* connector. Note that Sync, LossySync, and SyncDrain, in contrast, do not exhibit state-sensitivity (this also holds for Merger and Replicator), because they lack buffers. When depicting FIFO, we represent an empty buffer by a white box and a full buffer by a black box.

Merger routes a data item from one of its input nodes to its output node only if its output node has a pending take request and *at least* one of its input nodes has a pending write request. If both of its input nodes have a pending write request, Merger chooses one of these nodes non-deterministically. The chosen node then fires, while the write request on the other node remains pending.

Finally, Replicator routes a data item from its input node to both of its output nodes only if its input node has a pending write request and both of its output nodes have a pending take request. If Replicator has a pending take request on only one of its output nodes, it stays idle and all the I/O-requests remain pending.

*Composed connectors* The behavior of ExclRouter resembles that of Replicator, but with one important difference: whereas Replicator copies a data item dispatched at its input node to both of its output nodes, ExclRouter relays a data item to only one of its output nodes at a time. If both of its output nodes have a pending take request, ExclRouter chooses one of them non-deterministically and routes the data item to only that node. A more detailed discussion on ExclRouter appears in [Cos10].

LossyFIFO consists of two primitive connectors: LossySync and FIFO. Introduced in [CCA07], it has become the classical example for showing that a semantic model can (or cannot) describe the behavior of context-sensitive connectors. As its name suggests, LossyFIFO behaves as a lossy version of FIFO: in case of an empty buffer, LossyFIFO and FIFO behave indistinguishably, but in case of a full buffer, LossyFIFO loses any data item dispatched at its input node. In [CCA07], Clarke et al. elaborate on LossyFIFO and use it to explain why the semantic models that precede the 3-coloring model (introduced in that publication) cannot describe the behavior of context-sensitive connectors.

Finally, although its composition looks complex, SyncFIFO behaves indistinguishably from FIFO, except for the case in which it has an empty buffer and pending I/O-requests on both of its nodes: whereas FIFO in this situation routes each data item from its input node to its buffer (the take request remains pending), SyncFIFO routes each data item from its input node *past its buffer* immediately to its output node (both I/O-requests succeed, while the buffer remains empty). Because we can illustrate the concepts that involve composed connectors in this report with the simpler ExclRouter and LossyFIFO, we do not discuss SyncFIFO again until Section 4.4 (page 52).

**Coloring models** We start with coloring models [CCA07,Cos10] as the first semantic model we discuss. Coloring models work by marking nodes of a connector with *colors* that specify whether data items flow through these nodes or not. Depending on the number of colors, different coloring models with different levels of expressiveness arise. In this report, we assume a total of four colors and derive two classes of coloring models from this set: *2-coloring models* and *3-coloring models*.

**Definition 5 (Colors [CCA07]).** $\mathcal{C}\mathsf{olor} = \{$ ——, - - - -, - ⇾ -, - ⇽ - $\}$ *is a set of colors.* 2-$\mathcal{C}\mathsf{olor} \subset \mathcal{C}\mathsf{olor}$ *is a subset of colors defined as* $\{$ ——, - - - - $\}$. 3-$\mathcal{C}\mathsf{olor} \subset \mathcal{C}\mathsf{olor}$ *is a subset of colors defined as* $\{$ ——, - ⇾ -, - ⇽ - $\}$.

The *flow color* —— indicates that data items flow through the nodes it marks, whereas the *no-flow color* - - - - indicates the converse: no data items flow through the nodes it marks. Together, these two colors constitute the class of 2-coloring models, which many consider incapable of describing the behavior of context-sensitive connectors. The 3-coloring model, in contrast, consists of two no-flow colors (- ⇾ - and - ⇽ -) instead of only one. This allows us to model not only *that* data items cannot flow through a node, but also *why*. More precisely, in the 3-coloring model, the direction of the arrow of the no-flow colors indicates where the reason for the absence of flow comes from: an arrow pointing in the same direction as the flow indicates that a node has no pending write requests, while an arrow pointing in the opposite direction indicates that a node has no pending take requests. In text, we associate - ⇾ - with the former case and - ⇽ - with the latter.

To describe a single behavior alternative of a connector in a given state, we define *colorings*: maps from sets of nodes to sets of colors that assign to each node in the set a color that states whether this node fires (or not) in the behavior that the coloring describes. We collect all behavior alternatives of a connector in sets of colorings called *coloring tables*.

**Definition 6 (Coloring [CCA07]).** *Let* $N \subseteq \mathcal{N}\mathsf{ode}$. *A coloring* $c$ *over* $N$ *is a total map* $N \to \mathcal{C}\mathsf{olor}$. *A coloring is a 2-coloring (respectively, 3-coloring) iff its co-domain is* 2-$\mathcal{C}\mathsf{olor}$ *(respectively,* 3-$\mathcal{C}\mathsf{olor}$*).*

**Definition 7 (Coloring table [CCA07]).** *Let* $N \subseteq \mathcal{N}\mathsf{ode}$. *A coloring table* $T$ *over* $N$ *is a set* $\{ N \to \mathcal{C}\mathsf{olor} \}$ *of colorings over* $N$. *A coloring table is a 2-coloring table (respectively, 3-coloring table) iff it is a set of 2-colorings (respectively, 3-colorings).*

To illustrate the previous definitions, coloring tables of some of the common state-insensitive primitive connectors appear in Figure 3 (page 8) and Figure 4 (page 9); a detailed explanation of the meaning of these coloring tables appears in [Cos10].

To accommodate connectors that behave differently in different states (e.g., connectors with buffers), we use *coloring table maps* (CTM): maps from sets of *indexes* (representing the states of a connector) to sets of coloring tables (representing the allowed behaviors in these states).[5] Subsequently, to model

[5] In [Cos10], Costa calls coloring table maps *indexed sets of coloring tables*.

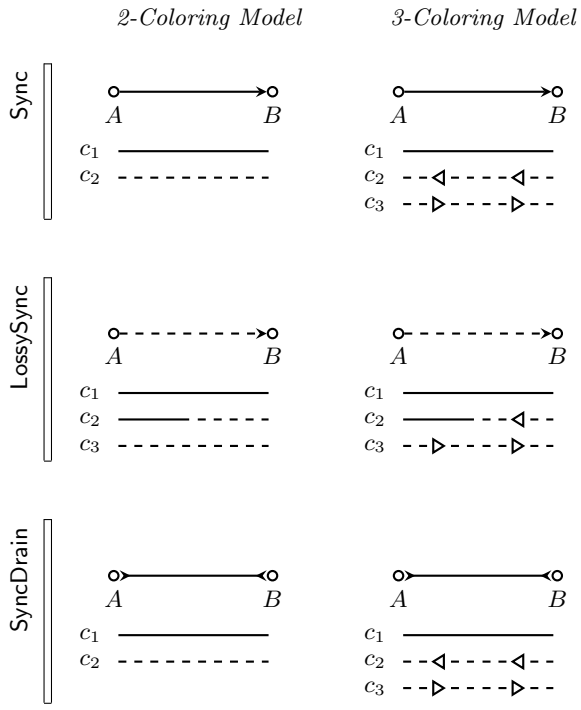


**Fig. 3.** Coloring models of Sync, LossySync, and SyncDrain.

the change of state a connector incurs when (some of) its nodes fire, we use *next functions*. The next function of a connector maps an index $\lambda$ in the domain $\Lambda$ of a CTM $S$ and a coloring in the coloring table to which $S$ maps $\lambda$ to some index in $\Lambda$ (possibly the same $\lambda$).

**Definition 8 (Universe of indexes).** *$\mathcal{I}$ndex is the set of indexes.*

**Definition 9 (Coloring table map [Cos10]).** *Let $N \subseteq \mathcal{N}$ode and $\Lambda \subseteq \mathcal{I}$ndex. A coloring table map $S$ over $[N, \Lambda]$ is a total map $S : \Lambda \to \{ N \to \mathcal{C}\text{olor} \}$ from indexes to coloring tables over $N$. A coloring table map is a 2-coloring table map (respectively, 3-coloring table map) iff its co-domain is $\{ N \to 2\text{-}\mathcal{C}\text{olor} \}$ (respectively, $\{ N \to 3\text{-}\mathcal{C}\text{olor} \}$).*

**Definition 10 (Next function [Cos10]).** *Let $S$ be a CTM over $[N, \Lambda]$. A next function $\eta$ over $[N, \Lambda, S]$ is a map $\Lambda \times \{ N \to \mathcal{C}\text{olor} \} \to \Lambda$ from [index, coloring]-pairs to indexes such that $[\lambda, c \mapsto \lambda'] \in \eta$ iff $c \in S(\lambda)$* (TOTALITY)*. It is a 2-colored next function (respectively, 3-colored next function) iff its domain is $\Lambda \times \{ N \to 2\text{-}\mathcal{C}\text{olor} \}$ (respectively, $\Lambda \times \{ N \to 3\text{-}\mathcal{C}\text{olor} \}$).*

To illustrate the previous definitions, in Figure 5 (page 10), we show the 2-colorings and 3-colorings of FIFO, its set of indexes $\Lambda$ for its two states EMPTY

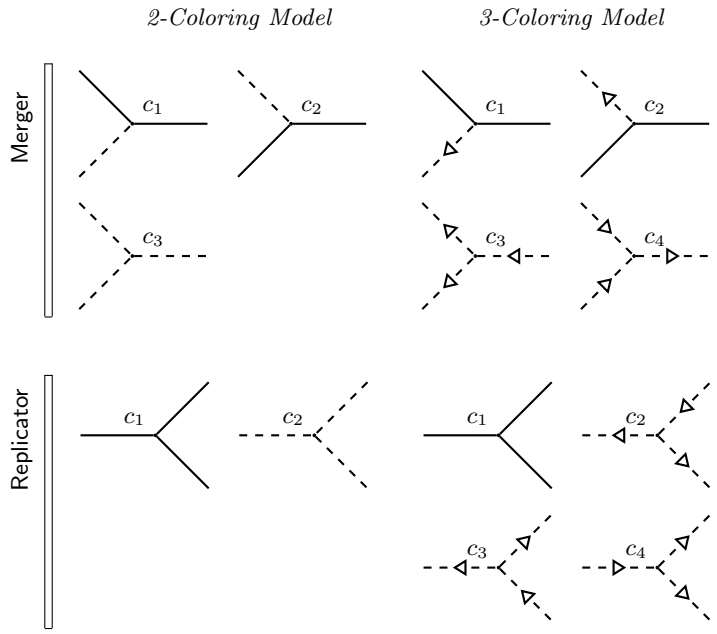


**Fig. 4.** Coloring models of Merger and Replicator.

and FULL (represented by the indexes `FIFO-E` and `FIFO-F`), its CTMs $S$, and its next functions $\eta$. Again, we refer to [Cos10] for a detailed explanation.

Finally, we join coloring models—i.e., next functions, which comprehensively describe the behavior of circuits—and connector structures in *$\eta$-connectors*: complete formal models of connectors.

**Definition 11 ($\eta$-connector).** *Let $S$ be a CTM over $[N, \Lambda]$. An $\eta$-connector $\mathcal{C}^{\mathsf{Col}}$ over $[N, \Lambda, S]$ is a tuple $\langle C, \eta \rangle$ in which $C = \langle B, E \rangle$ is a connector structure over $N$ and $\eta$ a next function over $[N, \Lambda, S]$. An $\eta$-connector is 2-colored (respectively, 3-colored) iff its constituent next function is a 2-colored next function (respectively, 3-colored next function).*

*Composition* When we compose two connectors that have coloring models as their formal semantics, we can compute the coloring model of the composed connector by composing the coloring models of its constituents. We describe this composition process in a bottom-up fashion. First, to compose two *compatible colorings*—i.e., colorings that assign the same colors to their shared nodes—we merge the domains of these colorings and map each node $n$ in the resulting set to the color that one of the colorings assigns to $n$.[6] The composition of two coloring

[6] Throughout this report, we implicitly apply the *flip-rule* [CCA07]. This rule states that if a coloring $c$ marks a node $n$ with a no-flow color whose arrow points towards $n$, the coloring identical to $c$ except that it marks $n$ with the *other* no-flow

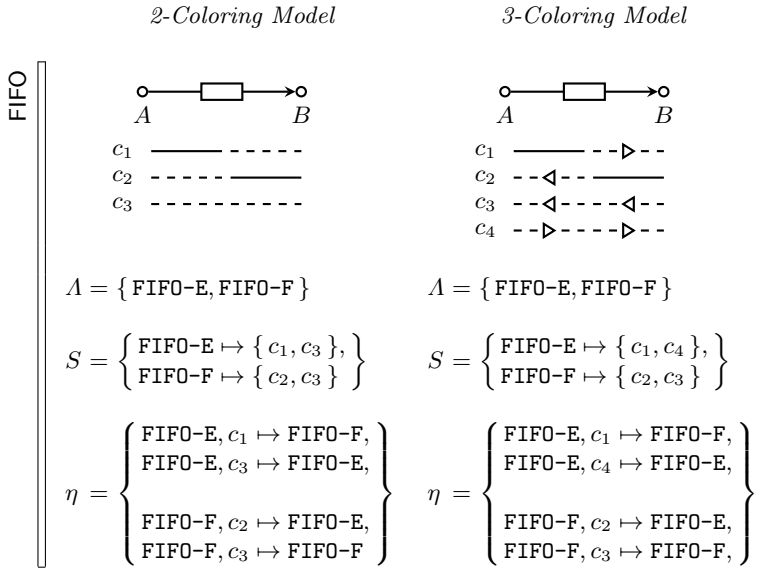


**Fig. 5.** Coloring models of FIFO.

tables then comprises the computation of a new coloring table that contains the pairwise compositions of the compatible colorings in the two individual coloring tables.

**Definition 12 (Composition of colorings [CCA07]).** *Let $c_1$ and $c_2$ be colorings over $N_1$ and $N_2$ such that $c_1(n) = c_2(n)$ for all $n \in N_1 \cap N_2$. Their composition, denoted $c_1 \cup c_2$, is a coloring over $N_1 \cup N_2$ defined as:*

$$c_1 \cup c_2 = \left\{ n \mapsto \kappa \;\middle|\; n \in N_1 \cup N_2 \text{ and } \kappa = \begin{pmatrix} c_1(n) \text{ if } n \in N_1 \\ c_2(n) \text{ otherwise} \end{pmatrix} \right\}$$

**Definition 13 (Composition of coloring tables [CCA07]).** *Let $T_1$ and $T_2$ be coloring tables over $N_1$ and $N_2$. Their composition, denoted $T_1 \cdot T_2$, is a coloring table over $N_1 \cup N_2$ defined as:*

$$T_1 \cdot T_2 = \left\{ c_1 \cup c_2 \;\middle|\; \begin{array}{l} c_1 \in T_1 \text{ and } c_2 \in T_2 \\ \text{and } c_1(n) = c_2(n) \text{ for all } n \in N_1 \cap N_2 \end{array} \right\}$$

To illustrate the previous definitions, in Figure 6 (page 11), we depict the 2-coloring tables and 3-coloring tables of ExclRouter, which we obtain by composing the coloring tables of its constituent primitive connectors. Rather than

color—i.e., the no-flow color with a *flipped* arrow—also describes an admissible behavior. This means that even if two colorings appear incompatible, after applying the flip-rule, two compatible colorings can arise. To keep coloring tables small, similar to [CCA07,Cos10], we do not include colorings that one can infer with the flip-rule in the coloring tables that we show.

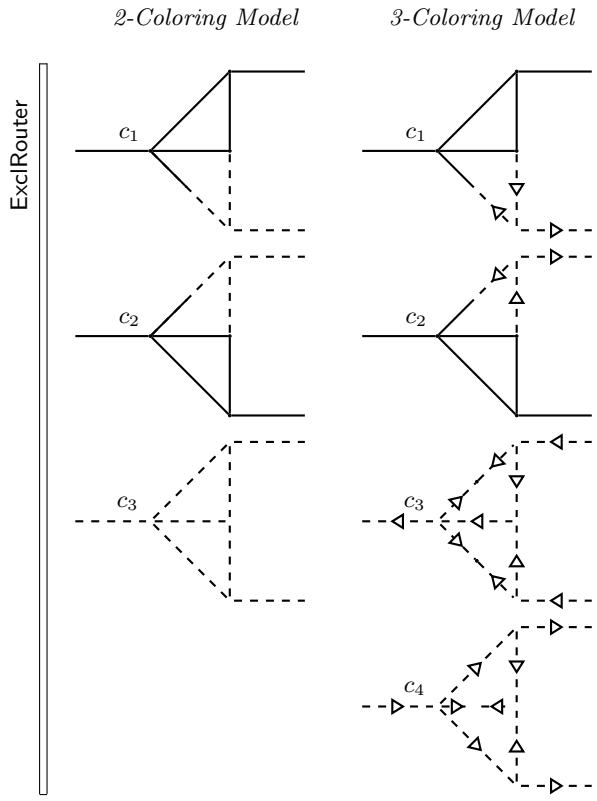


**Fig. 6.** Coloring models of ExclRouter.

depicting all the colorings that this composition process yields, we do not depict those colorings that contain *causality loops*. Causality loops may occur if a composed connector has one or more circular sub-circuits (as in the case of ExclRouter), and they can cause, among other anomalous phenomena, a reason for the absence of flow to appear out of nowhere. Colorings that contain causality loops, therefore, describe inadmissible behavior. In [Cos10], Costa proposes an algorithm for the detection and removal of colorings that contain causality loops. We tacitly apply this algorithm throughout this report.

Next, the composition of two CTMs comprises the computation of a new CTM that maps each pair of indexes in the Cartesian product of the domains of the two individual CTMs to the composition of the coloring tables to which these CTMs map the indexes in the pair. We define the composition of two next functions in terms of the Cartesian product, the composition of colorings, and the composition of CTMs.

**Definition 14 (Composition of CTMs [Cos10]).** *Let $S_1$ and $S_2$ be CTMs over $[N_1, \Lambda_1]$ and $[N_2, \Lambda_2]$. Their composition, denoted $S_1 \odot S_2$, is a CTM over*

$[N_1 \cup N_2, \Lambda_1 \times \Lambda_2]$ *defined as:*

$$S_1 \odot S_2 = \{\, \langle \lambda_1, \lambda_2 \rangle \mapsto S_1(\lambda_1) \cdot S_1(\lambda_2) \mid \lambda_1 \in \Lambda_1 \text{ and } \lambda_2 \in \Lambda_2 \,\}$$

**Definition 15 (Composition of next functions [Cos10]).** *Let $\eta_1$ and $\eta_2$ be next functions over $[N_1, \Lambda_1, S_1]$ and $[N_2, \Lambda_2, S_2]$. Their composition, denoted $\eta_1 \otimes \eta_2$, is a next function over $[N_1 \cup N_2, \Lambda_1 \times \Lambda_2]$ defined as:*

$$\eta_1 \otimes \eta_2 = \left\{ \begin{array}{c} \langle \lambda_1, \lambda_2 \rangle, c_1 \cup c_2 \\ \downarrow \\ \langle \eta_1(\lambda_1, c_1), \eta_2(\lambda_2, c_2) \rangle \end{array} \middle| \begin{array}{c} \langle \lambda_1, \lambda_2 \rangle \in \Lambda_1 \times \Lambda_2 \\ \text{and} \\ c_1 \cup c_2 \in (S_1 \odot S_2)(\langle \lambda_1, \lambda_2 \rangle) \end{array} \right\}$$

To illustrate the previous definitions, in Figure 7 (page 13), we depict the coloring models of LossyFIFO, which we obtain by composing the coloring models of LossySync and FIFO. In the figure, for notational convenience, we adopt the indexes `LosFIFO-E` and `LosFIFO-F` as aliases for the indexes $\langle$`LossySync`, `FIFO-E`$\rangle$ and $\langle$`LossySync`, `FIFO-F`$\rangle$. This example shows that the previously defined composition operators fail to properly compose 2-coloring models of context-sensitive connectors. To see this, first, we note that the 2-coloring $c_3$ of LossyFIFO describes a behavior in which LossyFIFO loses a data item. Next, we observe that the 2-CTM of LossyFIFO maps index `LosFIFO-E` to a set that contains $c_3$: this means that its 2-coloring model allows LossyFIFO to behave as described by $c_3$ in the EMPTY state—i.e., it may lose data items—despite the emptiness of the buffer. We consider this inadmissible behavior. In contrast, the 3-CTM of LossyFIFO maps `LosFIFO-E` to a set that excludes the 3-coloring $c_3$. Thus, whereas the 3-coloring model of LossyFIFO describes exactly the behavior that we intend it to exhibit, its 2-coloring model fails in this respect. As noted earlier, in recent years, the LossyFIFO connector has become the prime example for demonstrating that (and why) some formalisms cannot describe the behavior of context-dependent connectors.

Finally, we define the composition of $\eta$-connectors in terms of the composition of connector structures and the composition of next functions.

**Definition 16 (Composition of $\eta$-connectors [Cos10]).** *Let $\mathcal{C}_1^{\mathsf{Col}} = \langle C_1, \eta_1 \rangle$ and $\mathcal{C}_2^{\mathsf{Col}} = \langle C_2, \eta_2 \rangle$ be $\eta$-connectors over $[N_1, \Lambda_1, S_1]$ and $[N_2, \Lambda_2, S_2]$. Their composition, denoted $\mathcal{C}_1^{\mathsf{Col}} \times \mathcal{C}_2^{\mathsf{Col}}$, is an $\eta$-connector over $[N_1 \cup N_2, \Lambda_1 \times \Lambda_2, S_1 \odot S_2]$ defined as:*

$$\mathcal{C}_1^{\mathsf{Col}} \times \mathcal{C}_2^{\mathsf{Col}} = \langle C_1 \boxtimes C_2, \eta_1 \otimes \eta_2 \rangle$$

**Constraint automata** We end this section with *constraint automata* (CA) [BSAR06] as the second semantic model we discuss. A CA consists of a (possibly singleton) set of states, which correspond one-to-one to the states of the connector whose behavior it models, and a set of transitions between them; in contrast to standard automata, CA do not have accepting states. A transition of a CA carries a label that consists of two elements: a set of nodes and a *data constraint*. The former, called a *firing set*, describes which nodes can fire simultaneously in the state the transition leaves from; the latter specifies the conditions

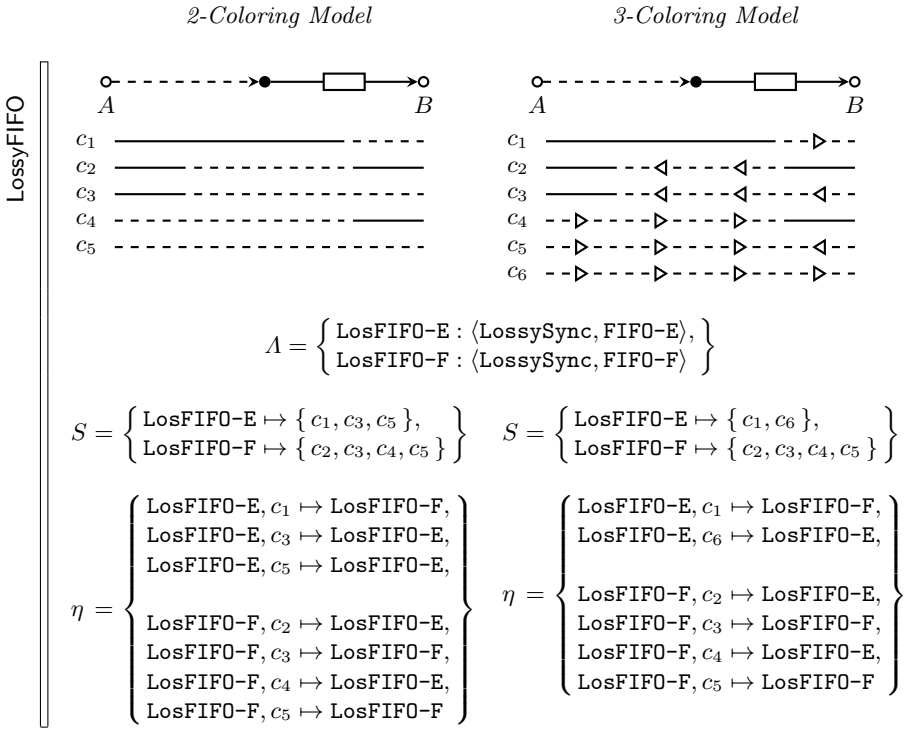


**Fig. 7.** Coloring models of LossyFIFO.

that the content of the data items that flow through these firing nodes must satisfy. Because we do not consider data constraints in this paper, we do not define their syntax and semantics here; instead, we often investigate a special class of CA, called *port automata* (PA), all of whose transitions carry the trivially satisfied constraint $\top$ [KC09]. Details on data constraints appear in [BSAR06]. To summarize the previous: CA serve as operational models of connector behavior, whose states correspond one-to-one to the states of a connector and whose transitions specify for each state when and what data items can flow through which nodes.

**Definition 17 (Universe of data constraints).** $\mathcal{C}\mathsf{onstraint}$ *is the set of data constraints (including* $\top$*).*

**Definition 18 (Constraint automaton [BSAR06]).** *Let* $N \subseteq \mathcal{N}\mathsf{ode}$ *and* $G \subseteq \mathcal{C}\mathsf{onstraint}$*. A constraint automaton* $\alpha$ *over* $[N, G]$ *is a tuple* $\langle Q, R, q_0 \rangle$ *in which* $Q$ *is a set of states,* $R \subseteq Q \times 2^N \times G \times Q$ *is a transition relation, and* $q_0 \in Q$ *is an initial state. A constraint automaton over* $[N, G]$ *is a port automaton over* $N$ *iff* $G = \{\top\}$*.*

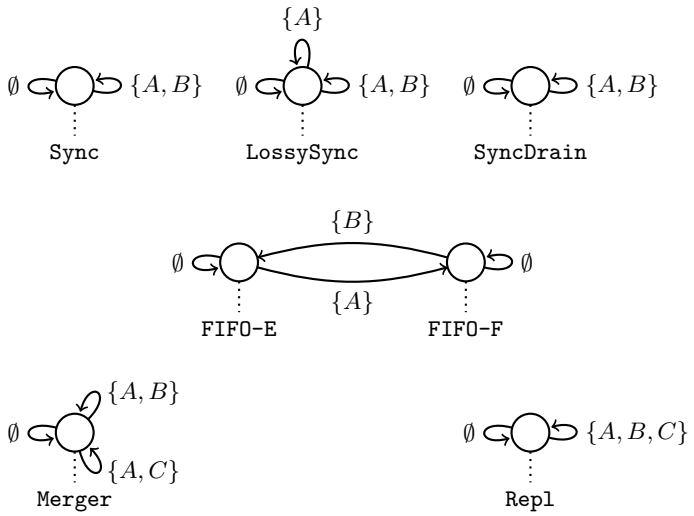


**Fig. 8.** Port automata of common primitives.

To illustrate the previous definitions, PA of some of the common primitive connectors appear in Figure 8 (page 14). Note that rather than naming states symbolically (e.g, $q, p, q_0, q_1, \ldots$), we name states by the same indexes that we encountered when defining coloring table maps of coloring models.

*Note 1.* In the sequel, if $\alpha = \langle Q, R, q_0 \rangle$ defines a CA then $Q \subseteq \mathcal{I}\mathsf{ndex}$.

Similar to next functions, CA comprehensively model the behavior of circuits. Analogous to $\eta$-connectors, therefore, we introduce $\alpha$-*connectors*: pairs that consist of a connector structure and a CA.

**Definition 19 ($\alpha$-connector).** *Let $N \subseteq \mathcal{N}\mathsf{ode}$ and $G \subseteq \mathcal{C}\mathsf{onstraint}$. An $\alpha$-connector $\mathcal{C}^{\mathsf{CA}}$ over $[N, G]$ is a pair $\langle C, \alpha \rangle$ in which $C = \langle N, B, E \rangle$ is a connector structure and $\alpha = \langle Q, R, q_0 \rangle$ is a CA over $[N, G]$.*

*Composition* When we compose two connectors that have CA as behavioral model, we can compute the CA of the composed connector by composing the CA of its constituents: the binary operator for CA composition takes the Cartesian product of the set of states of its arguments, designates the pair of their initial states as the initial state of the composed CA, and computes a new transition relation.

**Definition 20 (Composition of CA [BSAR06]).** *Let $\alpha_1 = \langle Q_1, R_1, q_0^1 \rangle$ and $\alpha_2 = \langle Q_2, R_2, q_0^2 \rangle$ be CA over $[N_1, G_1]$ and $[N_2, G_2]$. Their composition, denoted $\alpha_1 \bowtie \alpha_2$, is a CA over $[N_1 \cup N_2, G_1 \wedge G_2]$ defined as:*[7]

$$\alpha_1 \bowtie \alpha_2 = \langle Q_1 \times Q_2, R, \langle q_0^1, q_0^2 \rangle \rangle$$

[7] For notational convenience, we write $G_1 \wedge G_2$ for $\{ g_1 \wedge g_2 \mid g_1 \in G_1 \text{ and } g_2 \in G_2 \}$.

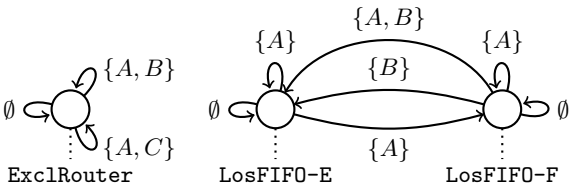


**Fig. 9.** Port automata of ExclRouter and LossyFIFO.

*in which:*

$$R = \left\{ \langle\langle q_1, q_2\rangle, F_1 \cup F_2, g_1 \wedge g_2, \langle q_1', q_2'\rangle\rangle \;\middle|\; \begin{array}{l} \langle q_1, F_1, g_1, q_1'\rangle \in R_1 \\ \textit{and } \langle q_2, F_2, g_2, q_2'\rangle \in R_2 \\ \textit{and } F_1 \cap N_2 = F_2 \cap N_1 \end{array} \right\}$$

The previous definition differs slightly from the one in [BSAR06]: we do not *implicitly* assume that every state have a silent $\tau$-transition (as in [BSAR06]), but *explicitly* include these transitions in our models (represented by transitions labeled with $\emptyset$). Though essentially a matter of representation, this simplifies later proofs. To illustrate the previous definition, in Figure 9 (page 15), we depict the PA of ExclRouter and LossyFIFO (for clarity, we hide internal nodes). Interestingly, when we disregard the names of the states, the transitions of the PA of ExclRouter equal the transitions of the PA of Merger. Furthermore, as its 2-coloring model in Figure 7 (page 13), the PA of LossyFIFO does not model the behavior that we intend LossyFIFO to exhibit: this automaton includes a transition, namely $\langle$LosFIFO-E$, \{A\}, \top,$ LosFIFO-E$\rangle$, which describes the same inadmissible behavior as coloring $c_3$ in the 2-coloring model of LossyFIFO (the loss of data items in the EMPTY state).

Similar to Definition 16 (page 12) of composition for $\eta$-connectors, we define the composition operator for $\alpha$-connectors in terms of the composition operators for connector structures and CA.

**Definition 21 (Composition of $\alpha$-connectors).** *Let $\mathcal{C}_1^{\mathsf{CA}} = \langle C_1, \alpha_1\rangle$ and $\mathcal{C}_2^{\mathsf{CA}} = \langle C_2, \alpha_2\rangle$ be $\alpha$-connectors over $[N_1, G_1]$ and $[N_2, G_2]$. Their composition, denoted $\mathcal{C}_1^{\mathsf{CA}} \times \mathcal{C}_2^{\mathsf{CA}}$, is an $\alpha$-connector over $[N_1 \cup N_2, G_1 \wedge G_2]$ defined as:*

$$\mathcal{C}_1^{\mathsf{CA}} \times \mathcal{C}_2^{\mathsf{CA}} = \langle C_1 \boxtimes C_2, \alpha_1 \bowtie \alpha_2\rangle$$

As a final remark, we emphasize that—more than our presentation of coloring models—our presentation of CA remains superficial: we covered only the essentials relevant to the rest of this report. A more comprehensive overview appears in [BSAR06].

## 3 From 3-colored $\eta$-connectors to 2-colored $\eta$-connectors

Many consider 2-coloring models incapable of describing the behavior of context-dependent connectors. In this section, however, we show that 2-coloring models

*can* describe these: at the expense of making the connector models more complex, we encode context-dependency using only two colors and without altering the existing composition operators. Essentially, we trade a more complex formalism—i.e., 3-coloring models—with simple connector models for a simpler formalism—i.e., 2-coloring models—with more complex ones. We achieve this with the $\mathbb{M}$-*transformation*, the topic of this section. The $\mathbb{M}$-transformation comprises a collection of unary operators, all denoted by $\mathbb{M}$ for notational convenience, that facilitate the transformation of a 3-colored $\eta$-connector to a corresponding 2-colored $\eta$-connector. This latter $\eta$-connector differs from the ordinary 2-colored $\eta$-connector that models the same communication medium: every 2-colored $\eta$-connector that results from composing $\mathbb{M}$-transformed 3-colored $\eta$-connectors describes only admissible behavior. In contrast, an $\eta$-connector that results from composing ordinary 2-colored $\eta$-connectors can, as demonstrated in the previous section, describe inadmissible behavior (e.g., the composition of LossySync and FIFO).

This section looks as follows. First, we give the definitions of $\mathbb{M}$ in Section 3.1 (page 16). Second, we prove its correctness in Section 3.2 (page 27). Third, we show that $\mathbb{M}$ distributes over composition in Section 3.3 (page 30). Finally, we discuss the inversion of $\mathbb{M}$ in Section 3.4 (page 36).

*Note 2.* In the sequel, we abbreviate "$\mathbb{M}$-transformed" by "$\mathbb{M}$-ed" (e.g., pronounced /emd/).

## 3.1 Definition of $\mathbb{M}$

The idea behind the $\mathbb{M}$-transformation arose from a remark made by Krause in his PhD thesis [Kra11]: he suggests that we can describe context-sensitive connectors with port automata by adding fictitious *negated nodes* to every connector model. The transformation operator that we define in this subsection follows this approach: it transforms a 3-colored $\eta$-connector $\mathcal{C}^{\mathsf{Col}}$ to a corresponding 2-colored $\eta$-connector by cloning all the nodes of $\mathcal{C}^{\mathsf{Col}}$ and by assigning *appropriate* colors (from 2-$\mathcal{C}$olor) to these clones. Recall from Definition 11 (page 9) that an $\eta$-connector consists of a connector structure and a next function. For the sake of separating concerns, we define $\mathbb{M}$ first for the structural aspects of $\eta$-connectors and second for their behavioral aspects.

**Structure** We start with the definition of $\mathbb{M}$ for sets of nodes. Informally, the application of $\mathbb{M}$ to a set of nodes yields a set that contains twice as many nodes: the original nodes, henceforth called *base nodes*, and their clones, henceforth called *context nodes*. Often, we call a base node and its corresponding context node each other's *duals*;[8] if $n$ denotes a base node, we denote its dual by $\overline{n}$.

[8] We disfavor the terminology of "negated nodes", because context nodes do not behave as the negation of base nodes: if data items flow through a base node, data items never flow through its dual, but if no data items flow through a base node, data items *not always* flow through its dual.

**Definition 22 ($\mathbb{M}$ for sets of nodes).** *Let $N \subseteq \mathcal{N}\mathsf{ode}$. Its $\mathbb{M}$-transformation, denoted $\mathbb{M}(N)$, is defined as:*

$$\mathbb{M}(N) = \bigcup_{n \in N} \{\, n, \overline{n} \,\}$$

*such that $\overline{n_1} \in \mathcal{N}\mathsf{ode}$ (WELL-F) and $[n_1 = n_2$ iff $\overline{n_1} = \overline{n_2}]$ (1-TO-1), for all $n_1, n_2 \in N$.*

The second side condition, 1-TO-1, states that each context nodes in an $\mathbb{M}$-ed set of nodes corresponds one-to-one to a base node in the same set—i.e., no two base nodes share the same context node, and no two context nodes share the same base node.

*Note 3.* In the sequel, we adopt the previous condition as a universal property of base nodes and context nodes. Furthermore, in the sequel, we refer to the 1-TO-1 condition in Definition 22 (page 17) (e.g., in proofs) even if the context of reference concerns single nodes rather than sets of nodes.

The following proposition states that the application of $\mathbb{M}$ to a set of nodes yields a well-formed set of nodes.

**Proposition 1 ($\mathbb{M}$-ed sets of nodes are sets of nodes).** *Let $N \subseteq \mathcal{N}\mathsf{ode}$. Then, $\mathbb{M}(N)$ is a set of nodes.*

*Proof. By the premise, $N \subseteq \mathcal{N}\mathsf{ode}$. Also, by* WELL-F *in Definition 22 (page 17), $\overline{n} \in \mathcal{N}\mathsf{ode}$, for all $n \in N$. Therefore, $\mathbb{M}(N)$ is a set of nodes.* □

We proceed with the definition of $\mathbb{M}$ for primitives. Similar to $\mathbb{M}$ for sets of nodes, the application of $\mathbb{M}$ to a primitive yields a primitive that consists of twice as many nodes: the original base nodes and their duals. Additionally, $\mathbb{M}$ reverses the direction of the flow through these duals—i.e., a base input node yields a context output node, whereas a base output node yields a context input node. This reversal facilitates the backwards propagation of the reason for the absence of flow through context nodes (similar to the no-flow color - -◁- -); shortly, we elaborate on this.

**Definition 23 ($\mathbb{M}$ for primitives).** *Let $e = (n_1^{j_1}, \ldots, n_k^{j_k})$ be a primitive over $N$. Its $\mathbb{M}$-transformation, denoted $\mathbb{M}(e)$, is defined as:*

$$\mathbb{M}(e) = (n_1^{j_1}, \ldots, n_k^{j_k}, \overline{n_1}^{\neg j_1}_{k+1}, \ldots, \overline{n_k}^{\neg j_k}_{2k})$$

*in which $\neg \mathrm{i} = \mathrm{o}$ and $\neg \mathrm{o} = \mathrm{i}$.*

The following proposition states that the application of $\mathbb{M}$ to a primitive yields a well-formed primitive.

**Proposition 2 ($\mathbb{M}$-ed primitives are primitives).** *Let $e = (n_1^{j_1}, \ldots, n_k^{j_k})$ be a primitive over $N$. Then, $\mathbb{M}(e)$ is a primitive over $\mathbb{M}(N)$.*

*Proof.* *By Definition 2 (page 3) of primitive, we must show* $\mathbb{M}(N) = \{ n_l \mid 1 \leq l \leq 2k \}$ (NODES), $j_i \in \{ \mathrm{i}, \mathrm{o} \}$ (IO), *and* $[i \neq i' \text{ iff } n_i \neq n_{i'}]$ (UNIQUENESS), *for all* $1 \leq i, i' \leq 2k$. *We start with* NODES. *Because* $e$ *is a primitive over* $N$ *by the premise,* $N = \{ n_l \mid 1 \leq l \leq k \}$. *Then, by Definition 22 (page 17) of* $\mathbb{M}$ *for sets of nodes,* $\mathbb{M}(N) = \{ n_l \mid 1 \leq l \leq 2k \}$. *Hence,* $\mathbb{M}(e)$ *satisfies* NODES. *We proceed with* IO. *Because* $e$ *is a primitive by the premise,* $j_i \in \{ \mathrm{i}, \mathrm{o} \}$ *for all* $1 \leq i, i' \leq k$. *Then, by Definition 23 (page 17) of* $\neg$, $j_i \in \{ \mathrm{i}, \mathrm{o} \}$ *for all* $1 \leq i, i' \leq 2k$. *Hence,* $\mathbb{M}(e)$ *satisfies* IO. *We end with* UNIQUENESS. *Because* $e$ *is a primitive by the premise,* $[i \neq i' \text{ iff } n_i \neq n_{i'}]$ *for all* $1 \leq i, i' \leq k$. *Then, by* 1-TO-1 *in Definition 22 (page 17),* $[i \neq i' \text{ iff } n_i \neq n_{i'}]$ *for all* $1 \leq i, i' \leq 2k$. *Hence,* $\mathbb{M}(e)$ *satisfies* UNIQUENESS. *Because* $\mathbb{M}(e)$ *satisfies* NODES, IO, *and* UNIQUENESS, *it is a primitive over* $\mathbb{M}(N)$. □

We continue with the definition of $\mathbb{M}$ for sets of primitives. Informally, the application of $\mathbb{M}$ to a set of primitives yields a set of $\mathbb{M}$-ed primitives.

**Definition 24 ($\mathbb{M}$ for sets of primitives).** *Let* $E$ *be a set of primitives. Its* $\mathbb{M}$*-transformation, denoted* $\mathbb{M}(E)$*, is defined as:*

$$\mathbb{M}(E) = \{ \mathbb{M}(e) \mid e \in E \}$$

The following proposition states that the application of $\mathbb{M}$ to a set of primitives yields a well-formed set of primitives.

**Proposition 3 ($\mathbb{M}$-ed sets of primitives are sets of primitives).** *Let* $E$ *be a set of primitives. Then,* $\mathbb{M}(E)$ *is a set of primitives.*

*Proof.* *Because* $E$ *is a set of primitives by the premise and because* $\mathbb{M}$*-ed primitives are primitives by Proposition 2 (page 17),* $\mathbb{M}(E)$ *is a set of primitives.* □

We finish this part with the definition of $\mathbb{M}$ for connector structures. Recall from Definition 3 (page 3) that connector structures consist of a set of boundary nodes and a set of primitives. This allows us to define $\mathbb{M}$ for connector structures in terms of $\mathbb{M}$ for sets of nodes and $\mathbb{M}$ for sets of primitives.

**Definition 25 ($\mathbb{M}$ for connector structures).** *Let* $C = \langle B, E \rangle$ *be a connector structure over* $N$. *Its* $\mathbb{M}$*-transformation, denoted* $\mathbb{M}(C)$*, is defined as:*

$$\mathbb{M}(C) = \langle \mathbb{M}(B), \mathbb{M}(E) \rangle$$

The following proposition states that the application of $\mathbb{M}$ to a connector structure yields a well-formed connector structure. In our proof, we refer to a lemma that concerns the distributivity of $\mathbb{M}$ for sets of nodes over the common set operators. We have not discussed this lemma yet and defer its treatise to Section 3.3 (page 30). In that subsection, we prove the distributivity properties of all the forms of $\mathbb{M}$ relevant to this report.

**Proposition 4 ($\mathbb{M}$-ed connector structures are connector structures).** *Let* $C = \langle B, E \rangle$ *be a connector structure over* $N$. *Then,* $\mathbb{M}(C)$ *is a connector structure over* $\mathbb{M}(N)$.

**Table 1.** Proof: $\mathbb{M}(N) = \bigcup_{e \in \mathbb{M}(E)} \{\, n \in N' \mid e \text{ is a primitive over } N' \,\}$

| |
|---|
| $\mathbb{M}(N)$ |
| $=$ /* By applying $N = \bigcup_{e \in E} \{\, n \in N' \mid e \text{ is a primitive over } N' \,\}$ */ |
| $\mathbb{M}\left(\bigcup_{e \in E} \{\, n \in N' \mid e \text{ is a primitive over } N' \,\}\right)$ |
| $=$ /* By the distributivity of $\mathbb{M}$ for sets of nodes over union in Lemma 6 (page 30) */ |
| $\bigcup_{e \in E} \mathbb{M}\left(\{\, n \in N' \mid e \text{ is a primitive over } N' \,\}\right)$ |
| $=$ /* By Definition 23 (page 17) of $\mathbb{M}$ for primitives and because $\mathbb{M}$-ed primitives are primitives by Proposition 2 (page 17) */ |
| $\bigcup_{e \in E} \{\, n \in N' \mid \mathbb{M}(e) \text{ is a primitive over } N' \,\}$ |
| $=$ /* Because $\mathbb{M}(C)$ satisfies PRIMS by Proposition 4 (page 18) */ |
| $\bigcup_{e \in \mathbb{M}(E)} \{\, n \in N' \mid e \text{ is a primitive over } N' \,\}$ |

*Proof.* *By Definition 3 (page 3) of connector structures, we must show that* $\emptyset \neq \mathbb{M}(B) \subseteq \mathbb{M}(N)$ (B-NODES), $\mathbb{M}(E)$ *is a set of primitives* (PRIMS), *and* $\mathbb{M}(N) = \bigcup_{e \in \mathbb{M}(E)} \{\, n \in N' \mid e \text{ is a primitive over } N' \,\}$ (NODES). *We start with* PRIMS. *Because* $C$ *is a connector structure over* $N$ *by the premise,* $E$ *is a set of primitives. Then, because* $\mathbb{M}$*-ed sets of primitives are sets of primitives by Proposition 3 (page 18),* $\mathbb{M}(E)$ *is a set of primitives. Hence,* $\mathbb{M}(C)$ *satisfies* PRIMS. *We proceed with* NODES. *Because* $C$ *is a connector structure over* $N$ *by the premise,* $N = \bigcup_{e \in E} \{\, n \in N' \mid e \text{ is a primitive over } N' \,\}$. *Subsequently,* $\mathbb{M}(N) = \bigcup_{e \in \mathbb{M}(E)} \{\, n \in N' \mid e \text{ is a primitive over } N' \,\}$ *follows from Table 1 (page 19). Hence,* $\mathbb{M}(C)$ *satisfies* NODES. *We end with* B-NODES. *Because* $C$ *is a connector structure over* $N$ *by the premise,* $\emptyset \neq B \subseteq N$. *Then, by Definition 22 (page 17) of* $\mathbb{M}$ *for sets of nodes,* $\emptyset \neq \mathbb{M}(B) \subseteq \mathbb{M}(N)$. *Hence,* $\mathbb{M}(C)$ *satisfies* B-NODES. *Because* $\mathbb{M}(C)$ *satisfies* B-NODES, PRIMS, *and* NODES, *it is a connector structure over* $\mathbb{M}(N)$. □

To illustrate the application of $\mathbb{M}$ to connector structures, in Figure 10 (page 20), Figure 11 (page 20), and Figure 12 (page 21), we depict the $\mathbb{M}$-transformations of the connector structures of some of the common primitive connectors (their formal definitions appear below), ExclRouter, and LossyFIFO.

$$\begin{aligned}
\mathsf{Sync}, \mathsf{LossySync}, \mathsf{FIFO} &\triangleq \langle \{\, A, B, \overline{A}, \overline{B} \,\}, \{\, (A^{\mathrm{i}}, B^{\mathrm{o}}, \overline{A}^{\mathrm{o}}, \overline{B}^{\mathrm{i}}) \,\} \rangle \\
\mathsf{SyncDrain} &\triangleq \langle \{\, A, B, \overline{A}, \overline{B} \,\}, \{\, (A^{\mathrm{i}}, B^{\mathrm{o}}, \overline{A}^{\mathrm{o}}, \overline{B}^{\mathrm{o}}) \,\} \rangle \\
\mathsf{Merger} &\triangleq \langle \{\, A, B, C \,\}, \{\, (A^{\mathrm{i}}, B^{\mathrm{i}}, C^{\mathrm{o}}, \overline{A}^{\mathrm{o}}, \overline{B}^{\mathrm{o}}, \overline{C}^{\mathrm{i}}) \,\} \rangle \\
\mathsf{Replicator} &\triangleq \langle \{\, A, B, C \,\}, \{\, (A^{\mathrm{i}}, B^{\mathrm{o}}, C^{\mathrm{o}}, \overline{A}^{\mathrm{o}}, \overline{B}^{\mathrm{i}}, \overline{C}^{\mathrm{i}}) \,\} \rangle
\end{aligned}$$

Although we have not discussed the semantics—i.e., the coloring models—that the $\mathbb{M}$-transformation assigns to these structures yet, their graphical representations give away some hints. For example, the depictions of the $\mathbb{M}$-ed structures of ExclRouter and LossyFIFO show that data items flow in the opposite direction through the context nodes when compared with the direction of the flow through

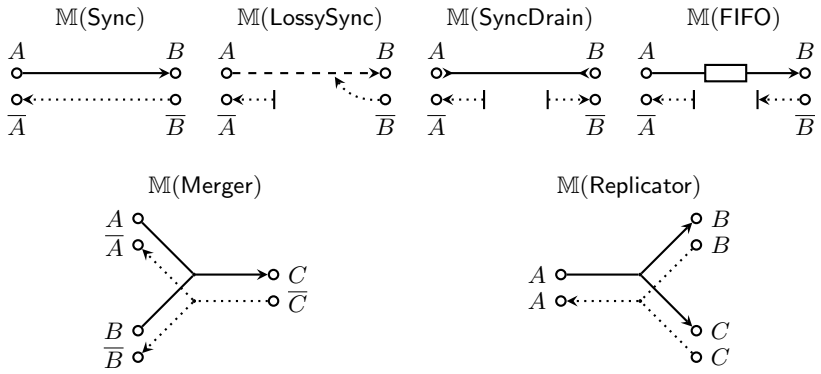


**Fig. 10.** Pictorial representation of the $\mathbb{M}$-ed versions of some of the common primitive connectors.

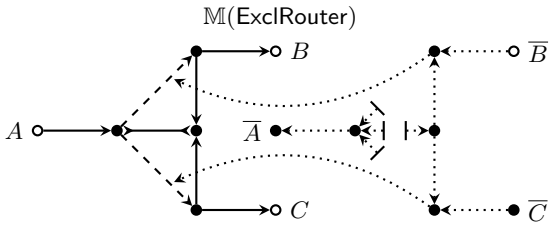


**Fig. 11.** Pictorial representation of the $\mathbb{M}$-ed version of ExclRouter.

the base nodes. As mentioned before, this resembles the way wherein the no-flow color - -⊲- - communicates the reason for the absence of flow backwards through a connector. Shortly, we demonstrate that flow through context nodes implies that data items can disappear between certain base nodes (conversely, the absence of flow through context nodes implies that data items cannot disappear).

Another—yet related—observation concerns the connections between base nodes and context nodes: they do not exist. More specifically, base nodes and context nodes form a *base circuit* and a *context circuit* that influence each other's behavior, but data items cannot flow from one of these circuits to the other. For instance, in Figure 11 (page 20) and Figure 12 (page 21), the new dotted arrows tangent to the dashed arrows, which represent the structure of LossySync, articulate the influence that a context circuit can exercise on its corresponding base circuit: informally, if a *context channel touches* a *base channel*, this base channel can lose data items only if data items flow through the context channel that touches it. If no context channels touch a base channel, this base channel cannot lose data items. In the following, we make the previous informal account more precise.

**Behavior** Recall from the introduction of this subsection that the application of $\mathbb{M}$ to a 3-colored $\eta$-connector $\mathcal{C}^{\mathsf{Col}}$ produces a corresponding 2-colored

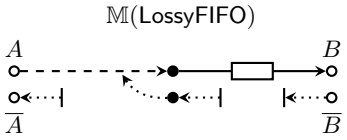


**Fig. 12.** Pictorial representation of the $\mathbb{M}$-ed version of LossyFIFO.

$\eta$-connector by cloning the nodes of $\mathcal{C}^{\mathsf{Col}}$ and by assigning appropriate colors to the resulting clones. In the foregoing, we discussed the cloning of nodes; in the following, we elaborate on the assignment of appropriate colors.

We begin with the definition of $\mathbb{M}$ for 3-colorings and clarify this definition afterwards.

**Definition 26 ($\mathbb{M}$ for 3-colorings).** *Let $c$ be a 3-coloring over $N$. Its $\mathbb{M}$-transformation, denoted $\mathbb{M}(c)$, is defined as:*

$$\mathbb{M}(c) = \bigcup_{n \in N} \begin{cases} \{ n \mapsto \texttt{- - - -}, \overline{n} \mapsto \texttt{- - - -} \} \text{ if } c(n) = \texttt{- -▷◁- -} \\ \{ n \mapsto \texttt{- - - -}, \overline{n} \mapsto \text{——} \} \text{ if } c(n) = \texttt{- -◁- -} \\ \{ n \mapsto \text{——}, \overline{n} \mapsto \texttt{- - - -} \} \text{ if } c(n) = \text{——} \end{cases}$$

Thus, the application of $\mathbb{M}$ to a 3-coloring $c$ yields a map from the nodes in the domain of $c$ and their duals to either —— or - - - -. The idea behind these mappings follows below.

- If $c$ maps $n$ to ——, $\mathbb{M}(c)$ also maps $n$ to ——, while it maps $\overline{n}$ to - - - -. This ensures that data items never flow through the same parts of the base and the context circuits synchronously. If we would allow such synchronous flow, for instance, data items could flow between the base nodes and through the context circuit of a LossySync (thus, this LossySync has pending write and take requests) at the same time. This would mean, however, that this LossySync may lose the data items that flow through its base circuit *without reason* (because of the pending take request). We consider this inadmissible behavior.
- If $c$ maps $n$ to - -◁- - (the no-flow color indicating that $n$ lacks take requests), $\mathbb{M}(c)$ maps $n$ to - - - - (because flow cannot appear out of nowhere), while it maps $\overline{n}$ to —— (because the absence of take requests can cause the loss of data items).
- If $c$ maps $n$ to - -▷◁- - (the no-flow color indicating that $n$ lacks write requests), $\mathbb{M}(c)$ maps $n$ to - - - - (because flow cannot appear out of nowhere), and the same holds for $\overline{n}$ (because the absence of write requests can never cause the loss of data items).

The following proposition states that the application of $\mathbb{M}$ to a 3-coloring yields a well-formed 2-coloring.

**Proposition 5 ($\mathbb{M}$-ed 3-colorings are 2-colorings).** *Let $c$ be a 3-coloring over $N$. Then, $\mathbb{M}(c)$ is a 2-coloring over $\mathbb{M}(N)$.*

*Proof.* *By Definition 6 (page 7) of 2-colorings, we must show that* $\mathbb{M}(c)$ *is a total map from* $\mathbb{M}(N)$ *to* 2-Color. *Because* $c$ *is a total map from* $N$ *to* 3-Color *by the premise and by Definition 26 (page 21) of* $\mathbb{M}$ *for 3-colorings,* $\mathbb{M}(c)$ *contains mappings for* $n$ *and* $\overline{n}$, *for all* $n \in N$. *Hence, by Definition 22 (page 17) of* $\mathbb{M}$ *for sets of nodes,* $\mathbb{M}(c)$ *is a total map with* $\mathbb{M}(N)$ *as its domain. Also, by Definition 26 (page 21) of* $\mathbb{M}$ *for 3-colorings,* $\mathbb{M}(c)$ *maps every* $n \in \mathbb{M}(N)$ *to either* ——— *or* - - - -. *Hence, by Definition 5 (page 7) of* 2-Color, $\mathbb{M}(c)$ *is a total map with* 2-Color *as its co-domain. Therefore,* $\mathbb{M}(c)$ *is a 2-coloring over* $\mathbb{M}(N)$. □

We proceed with the definition of $\mathbb{M}$ for 3-coloring tables. Recall that we defined a coloring table as a set of colorings. Informally, the application of $\mathbb{M}$ to a 3-coloring table—i.e., a set of 3-colorings—yields a set of $\mathbb{M}$-ed 3-colorings.

**Definition 27 ($\mathbb{M}$ for 3-coloring tables).** *Let* $T$ *be a 3-coloring table over* $N$. *Its* $\mathbb{M}$*-transformation, denoted* $\mathbb{M}(T)$, *is defined as:*

$$\mathbb{M}(T) = \{\, \mathbb{M}(c) \mid c \in T \,\}$$

The following proposition states that the application of $\mathbb{M}$ to a 3-coloring table yields a well-formed 2-coloring table.

**Proposition 6 ($\mathbb{M}$-ed 3-coloring tables are 2-coloring tables).** *Let* $T$ *be a 3-coloring table over* $N$. *Then,* $\mathbb{M}(T)$, *is a 2-coloring table over* $\mathbb{M}(N)$.

*Proof.* *Because* $T$ *is a set of 3-colorings over* $N$ *by the premise and because* $\mathbb{M}$*-ed 3-colorings are 2-colorings by Proposition 5 (page 21),* $\mathbb{M}(T)$ *is a set of 2-colorings over* $\mathbb{M}(N)$. □

To illustrate the application of $\mathbb{M}$ to 3-coloring tables, in Figure 13 (page 23), Figure 14 (page 23), and Figure 15 (page 24), we depict the $\mathbb{M}$-ed 3-coloring tables of some of the common primitive connectors and ExclRouter.[9] Although most transformations turn out straightforwardly, we observe the following about the transformation of the 3-coloring table of Merger. From a structural point of view, the context circuit that constitutes the $\mathbb{M}$-ed connector structure of Merger equals the ordinary connector structure of Replicator: both consist of one input node and two output nodes. The 2-colorings of this context circuit in Figure 14 (page 23), however, differ significantly from the 2-colorings in the ordinary 2-coloring table of Replicator: whereas the latter table does contain $c_3$ and $c_4$, it does not contain $c_1$ and $c_2$. Essentially, this means that context nodes as defined in this report do *not* have the same merger/replicator semantics as the type of nodes Arbab describes in [Arb04]. Note that we can observe a similar phenomenon when studying the $\mathbb{M}$-ed 3-coloring table of Replicator and the ordinary 2-coloring table of Merger.

We continue with the definition of $\mathbb{M}$ for 3-CTMs. Recall from Definition 9 (page 8) that a CTM maps indexes (representing the states of a connector) to

[9] We do not depict the $\mathbb{M}$-transformations of 3-colorings that one can induce with the flip-rule. See also Footnote 6 (page 9).

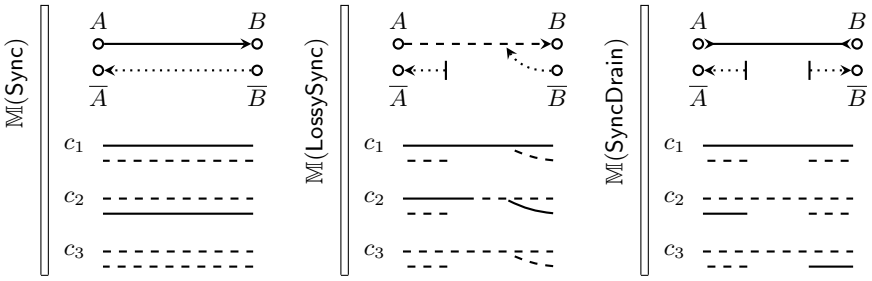


**Fig. 13.** 𝕄-ed 3-coloring models of Sync, LossySync, and SyncDrain.

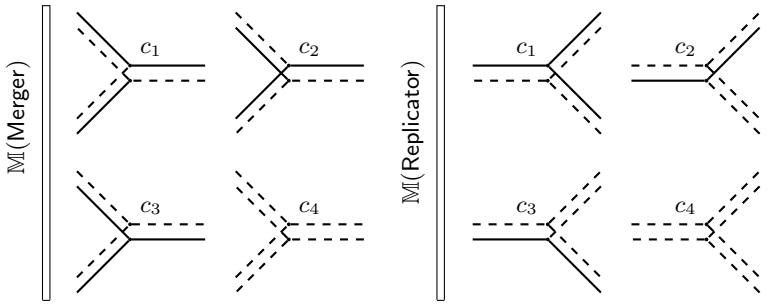


**Fig. 14.** 𝕄-ed 3-coloring models of Merger and Replicator.

coloring tables (describing the admissible behavior in these states). Informally, the application of 𝕄 to a 3-CTM yields a 2-CTM that maps indexes to 𝕄-ed 3-coloring tables.

**Definition 28 (𝕄 for 3-CTMs).** *Let $S$ be a CTM over $[N, \Lambda]$. Its 𝕄-transformation, denoted $\mathbb{M}(S)$, is defined as:*

$$\mathbb{M}(S) = \{\, \lambda \mapsto \mathbb{M}(S(\lambda)) \mid \lambda \in \Lambda \,\}$$

The following proposition states that the application of 𝕄 to a 3-CTM yields a well-formed 2-CTM.

**Proposition 7 (𝕄-ed 3-CTMs are 2-CTMs).** *Let $S$ be a 3-CTM over $[N, \Lambda]$. Then, $\mathbb{M}(S)$ is a 2-CTM over $[\mathbb{M}(N), \Lambda]$.*

*Proof.* *By Definition 9 (page 8) of 2-CTMs, we must show that $\mathbb{M}(S)$ is a total map from $\Lambda$ to a set of 2-coloring tables over $\mathbb{M}(N)$. Because $S$ is a total map from $\Lambda$ to a set of 3-coloring tables over $N$ by the premise and by Definition 28 (page 23) of 𝕄 for 3-CTMs, $\mathbb{M}(S)$ is a total map with $\Lambda$ as its domain. Also, because 𝕄-ed 3-coloring tables are 2-coloring tables by Proposition 6 (page 22), $\mathbb{M}(S)$ maps every $\lambda \in \Lambda$ to an 𝕄-ed 2-coloring table over $\mathbb{M}(N)$. Hence, $\mathbb{M}(S)$ is a total map with $\Lambda$ as its domain and with a set of 2-coloring tables over $\mathbb{M}(N)$ as its co-domain. Therefore, $\mathbb{M}(S)$ is a 2-coloring table map over $[\mathbb{M}(N), \Lambda]$.* □

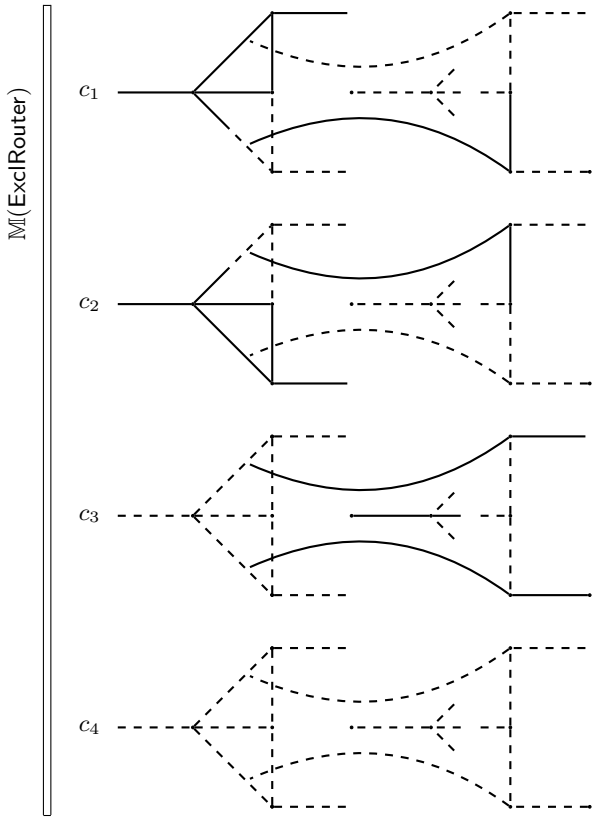


**Fig. 15.** $\mathbb{M}$-ed 3-coloring model of ExclRouter.

We proceed with the definition of $\mathbb{M}$ for 3-colored next functions. Recall that a next function maps [index, coloring]-pairs to indexes. Informally, the application of $\mathbb{M}$ to a 3-colored next function yields a 2-colored next function that maps [index, $\mathbb{M}$-ed 3-coloring]-pairs to indexes.

**Definition 29 ($\mathbb{M}$ for 3-colored next functions).** *Let $\eta$ be a 3-colored next function over $[N, \Lambda, S]$. Its $\mathbb{M}$-transformation, denoted $\mathbb{M}(\eta)$, is defined as:*

$$\mathbb{M}(\eta) = \{\, (\lambda, \mathbb{M}(c)) \mapsto \eta(\lambda, c)) \mid \lambda \in \Lambda \text{ and } c \in S(\lambda) \,\}$$

The previous definition may surprise the reader: it seems as if we do not apply the $\mathbb{M}$-transformation for 3-CTMs as one might expect. The following proposition establishes, however, that by defining $\mathbb{M}$ for 3-colored next functions as in the previous definition, we apply $\mathbb{M}$ to 3-CTMs rather *implicitly*: the proposition states that the application of $\mathbb{M}$ to a 3-colored next function over a set of nodes, a set of indexes, and a 3-CTM yields a well-formed 2-colored next function over the $\mathbb{M}$-ed set of nodes, the same set of indexes, *and the $\mathbb{M}$-ed 3-CTM.*

**Table 2.** Proof: $[\lambda, c \mapsto \lambda'] \in \mathbb{M}(\eta)$ iff $c \in (\mathbb{M}(S))(\lambda)$

| |
|---|
| $[\lambda, c \mapsto \lambda'] \in \mathbb{M}(\eta)$ |
| $\equiv$ /* By Definition 29 (page 24) of $\mathbb{M}$ for 3-colored next functions */ |
| there exists a $c' \in S(\lambda)$ such that $c = \mathbb{M}(c')$ |
| $\equiv$ /* Because, by Definition 27 (page 22) of $\mathbb{M}$ for 3-coloring tables, $c' \in S(\lambda)$ iff $\mathbb{M}(c') \in \mathbb{M}(S(\lambda))$ */ |
| there exists a $c' \in S(\lambda)$ such that $c = \mathbb{M}(c')$ and $\mathbb{M}(c') \in \mathbb{M}(S(\lambda))$ |
| $\equiv$ /* By simplifying $[c = \mathbb{M}(c')$ and $\mathbb{M}(c') \in \mathbb{M}(S(\lambda)]$ */ |
| there exists a $c' \in S(\lambda)$ such that $c \in \mathbb{M}(S(\lambda))$ |
| $\equiv$ /* Because $c'$ is a dead variable */ |
| $c \in \mathbb{M}(S(\lambda))$ |
| $\equiv$ /* By Definition 28 (page 23) of $\mathbb{M}$ for 3-CTMs */ |
| $c \in (\mathbb{M}(S))(\lambda)$ |

**Proposition 8 ($\mathbb{M}$-ed 3-colored next functions are 2-colored next functions).** *Let $\eta$ be a 3-colored next function over $[N, \Lambda, S]$. Then, $\mathbb{M}(\eta)$ is a 2-colored next function over $[\mathbb{M}(N), \Lambda, \mathbb{M}(S)]$.*

*Proof.* *By Definition 10 (page 8) of 2-colored next functions, we must show that $\mathbb{M}(\eta)$ is a map from [index, 2-coloring]-pairs to indexes such that $[\lambda, c \mapsto \lambda'] \in \mathbb{M}(\eta)$ iff $c \in (\mathbb{M}(S))(\lambda)$* (TOTALITY). *Because $\eta$ is a map from [index, 3-coloring]-pairs to indexes by the premise, by Definition 29 (page 24) of $\mathbb{M}$ for 3-colored next functions and because $\mathbb{M}$-ed 3-colorings are 2-colorings by Proposition 5 (page 21), $\mathbb{M}(\eta)$ is a map with $\Lambda \times \{\, \mathbb{M}(N) \to 2\text{-}\mathsf{Color} \,\}$ as its domain and with $\Lambda$ as its co-domain. Furthermore, satisfaction of* TOTALITY *by $\mathbb{M}(\eta)$ follows from Table 2 (page 25). Therefore, $\mathbb{M}(\eta)$ is a 2-colored next function over $[\mathbb{M}(N), \Lambda, \mathbb{M}(S)]$.* □

To illustrate the application of $\mathbb{M}$ to 3-colored next functions, we show the $\mathbb{M}$-ed 3-colorings, $\mathbb{M}$-ed 3-CTMs, and $\mathbb{M}$-ed 3-colored next functions of FIFO and LossyFIFO in Figure 16 (page 26).[10]

We end this subsection with the definition of $\mathbb{M}$ for 3-colored $\eta$-connectors. Recall from Definition 11 (page 9) that $\eta$-connectors consist of a connector structure and a next function. This allows us to define $\mathbb{M}$ for 3-colored $\eta$-connectors in terms of $\mathbb{M}$ for connector structures and $\mathbb{M}$ for 3-colored next functions.

**Definition 30 ($\mathbb{M}$ for 3-colored $\eta$-connectors).** *Let $\mathcal{C}^{\mathsf{Col}} = \langle C, \eta \rangle$ be a 3-colored $\eta$-connector over $[N, \Lambda, S]$. Its $\mathbb{M}$-transformation, denoted $\mathbb{M}(\mathcal{C}^{\mathsf{Col}})$, is defined as:*

$$\mathbb{M}(\mathcal{C}^{\mathsf{Col}}) = \langle \mathbb{M}(C), \mathbb{M}(\eta) \rangle$$

[10] We do not depict the $\mathbb{M}$-transformations of 3-colorings that one can induce with the flip-rule. See also Footnote 6 (page 9).

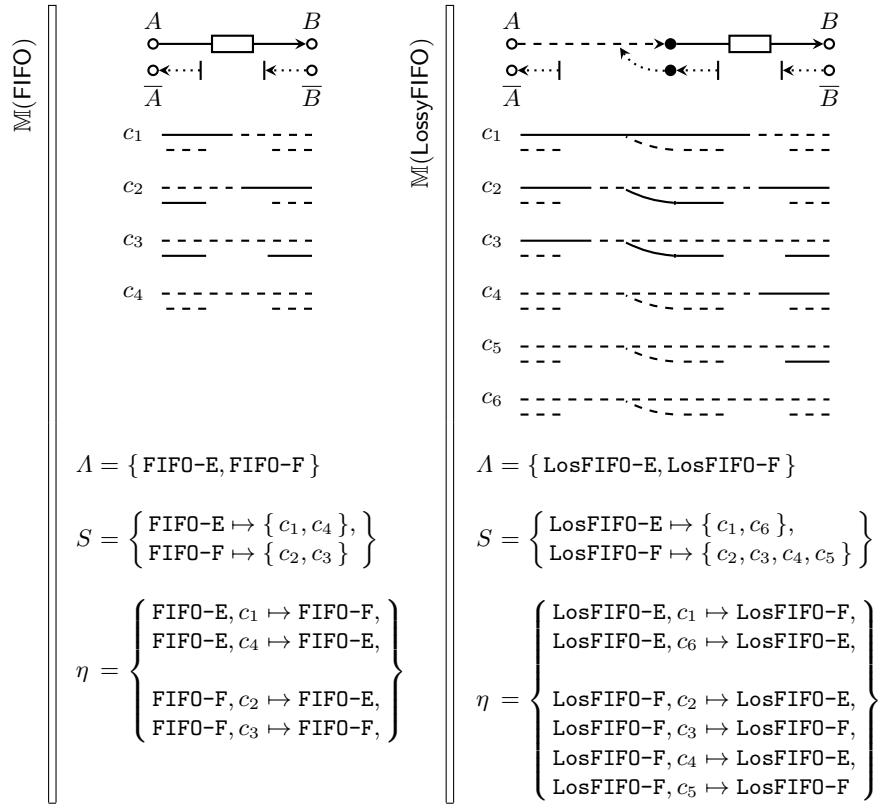


**Fig. 16.** $\mathbb{M}$-ed 3-coloring models of FIFO and LossyFIFO.

The following proposition states that the application of $\mathbb{M}$ to a 3-colored $\eta$-connector yields a well-formed 2-colored $\eta$-connector.

**Proposition 9 ($\mathbb{M}$-ed 3-colored $\eta$-connectors are 2-colored $\eta$-connectors).** *Let $\mathcal{C}^{\mathsf{Col}} = \langle C, \eta \rangle$ be a 3-colored $\eta$-connector over $[N, \Lambda, S]$. Then, $\mathbb{M}(\mathcal{C}^{\mathsf{Col}})$ is a 2-colored $\eta$-connector over $[\mathbb{M}(N), \Lambda, \mathbb{M}(S)]$.*

*Proof.* *By Definition 11 (page 9) of 2-colored $\eta$-connectors, we must show that $\mathbb{M}(C)$ is a connector structure over $\mathbb{M}(N)$ and that $\mathbb{M}(\eta)$ is a 2-colored next function over $[\mathbb{M}(N), \Lambda, \mathbb{M}(S)]$. Because $\mathcal{C}^{\mathsf{Col}}$ is a 3-colored $\eta$-connector over $[N, \Lambda, S]$ by the premise and by Definition 11 (page 9) of $\eta$-connectors, $C$ is a connector structure over $N$, and $\eta$ is a 3-colored next function over $[N, \Lambda, S]$. Then, because $\mathbb{M}$-ed connector structures are connector structures by Proposition 4 (page 18), $\mathbb{M}(C)$ is a connector structure over $\mathbb{M}(N)$. Also, because $\mathbb{M}$-ed 3-colored next functions are 2-colored next functions by Proposition 8 (page 25), $\mathbb{M}(\eta)$ is a 2-colored next function over $[\mathbb{M}(N), \Lambda, \mathbb{M}(S)]$.* □

## 3.2 Correctness of $\mathbb{M}$

In this subsection, we demonstrate the *correctness* of $\mathbb{M}$: we consider $\mathbb{M}$ correct if its application to a 3-colored $\eta$-connector yields a *corresponding* 2-colored $\eta$-connector. We investigate two types of correspondence, *painting correspondence* and *simulation*, and show that $\mathbb{M}$ for 3-colored $\eta$-connectors satisfies both.

**Painting correspondence** We start with painting correspondence. In this context, we call the $\mathbb{M}$-transformation correct if, for each 3-colored $\eta$-connector $\mathcal{C}^{\mathsf{Col}}$, its $\mathbb{M}$-transformation $\mathbb{M}(\mathcal{C}^{\mathsf{Col}})$ has *paintings* that correspond to the paintings of $\mathcal{C}^{\mathsf{Col}}$. Informally, a painting comprises an (infinite) sequence of alternating indexes (representing states) and colorings (representing transitions); one may consider paintings as *executions* of $\eta$-connectors.

**Definition 31 (Painting).** *Let $\mathcal{C}^{\mathsf{Col}} = \langle C, \eta \rangle$ be an $\eta$-connector and $\lambda_0 \in \Lambda$ the index representing the initial state of the connector that $\mathcal{C}^{\mathsf{Col}}$ models. A painting of $\mathcal{C}^{\mathsf{Col}}$ is a sequence $[\lambda_0, c_0, \ldots]$ such that $\lambda_{i+1} = \eta(\lambda_i, c_i)$ for all $i \geq 0$. The set that contains all the paintings of $\mathcal{C}^{\mathsf{Col}}$ is denoted $\pi_{\mathcal{C}^{\mathsf{Col}}}$.*

Informally, two paintings correspond if, for all $i \geq 0$, the $i$-th colorings on the paintings assign the flow color to the same nodes in their domain intersection and the $i$-th indexes on the paintings equal each other. We formulate our correctness lemma formally below; a proof follows shortly.

**Lemma 1 ($\mathbb{M}$-ed 3-colored $\eta$-connectors correspond to 3-colored $\eta$-connectors).** *Let $\mathcal{C}^{\mathsf{Col}} = \langle C, \eta \rangle$ be a 3-colored $\eta$-connector over $[N, \Lambda, S]$ and $\lambda_0 \in \Lambda$ the index representing the initial state of the connector that $\mathcal{C}^{\mathsf{Col}}$ models. Then:*

I. *if:* $[\lambda_0, c_0, \ldots] \in \pi_{\mathcal{C}^{\mathsf{Col}}}$
*then:* $[\lambda_0, \mathbb{M}(c_0), \ldots] \in \pi_{\mathbb{M}(\mathcal{C}^{\mathsf{Col}})}$ *such that for all* $0 \geq i$ :
$\{\, n \in N \mid c_i(n) = \text{——} \,\} = \{\, n \in N \mid (\mathbb{M}(c_i))(n) = \text{——} \,\}$

II. *if:* $[\lambda_0, \mathbb{M}(c_0), \ldots] \in \pi_{\mathbb{M}(\mathcal{C}^{\mathsf{Col}})}$
*then:* $[\lambda_0, c_0, \ldots] \in \pi_{\mathcal{C}^{\mathsf{Col}}}$ *such that for all* $0 \geq i$ :
$\{\, n \in N \mid c_i(n) = \text{——} \,\} = \{\, n \in N \mid (\mathbb{M}(c_i))(n) = \text{——} \,\}$

Shortly, we sketch a proof by induction that establishes the lemma. For the sake of conciseness, however, we first move large parts of the inductive step to the following two lemmas. Lemma 2 (page 27) states that $\mathbb{M}$ for 3-colored next functions preserves the flow behavior of the connector—i.e., if an untransformed coloring assigns flow to some base node $n$, the $\mathbb{M}$-transformation of this coloring exists and assigns the flow color to $n$. The same must hold in the opposite direction. Lemma 3 (page 28) states that $\mathbb{M}$ for next functions preserves transitions from one state to the next.

**Lemma 2 ($\mathbb{M}$ for 3-colored next functions preserves flow).** *Let $\eta$ be a 3-colored next function over $[N, \Lambda, S]$ and $n \in N$. Then:*

$$\begin{pmatrix} \lambda \in \Lambda \text{ and } c \in S(\lambda) \\ \text{and } c(n) = \text{——} \end{pmatrix} \text{ iff } \begin{pmatrix} \lambda \in \Lambda \text{ and } \mathbb{M}(c) \in (\mathbb{M}(S))(\lambda) \\ \text{and } (\mathbb{M}(c))(n) = \text{——} \end{pmatrix}$$

*Proof. We first prove the left-to-right direction (*ONLY IF*), and proceed with the right-to-left direction (*IF*).*

ONLY IF — *We start by deriving the first two conjuncts of the right-hand side (RHS) from the first two conjuncts of the left-hand side (LHS). This turns out straightforwardly: the first conjunct follows trivially, while $c \in S(\lambda)$ implies $\mathbb{M}(c) \in (\mathbb{M}(S))(\lambda)$ by Definition 27 (page 22) of $\mathbb{M}$ for 3-coloring tables. Finally, we derive the RHS's third conjunct from the third conjunct of the LHS. By the premise, $c(n) =$ ——. Then, by Definition 26 (page 21) of $\mathbb{M}$ for 3-colorings, $\{ n \mapsto$ ——$, \overline{n} \mapsto$ - - - - $\} \subseteq \mathbb{M}(c)$. Hence, $(\mathbb{M}(c))(n) =$ ——.*

IF — *The first two conjuncts of the LHS follow from the first two conjuncts of the RHS similar to the* ONLY IF *case. Next, by the premise, $(\mathbb{M}(c))(n) =$ ——. By Definition 26 (page 21) of $\mathbb{M}$ for 3-colorings, this happens only if $c(n) =$ ——.* □

**Lemma 3 ($\mathbb{M}$ for 3-colored next functions preserves transitions).** *Let $\eta$ be a 3-colored next function over $[N, \Lambda, S]$ and $n \in N$. Then:*

$$\begin{pmatrix} \lambda, \lambda' \in \Lambda \text{ and } c \in S(\lambda) \\ \text{and } \eta(\lambda, c) = \lambda' \end{pmatrix} \text{ iff } \begin{pmatrix} \lambda, \lambda' \in \Lambda \text{ and } \mathbb{M}(c) \in (\mathbb{M}(S))(\lambda) \\ \text{and } (\mathbb{M}(\eta))(\lambda, \mathbb{M}(c)) = \lambda' \end{pmatrix}$$

*Proof. The implication, in both directions, follows trivially (first conjunct), from Definition 27 (page 22) of $\mathbb{M}$ for 3-coloring tables (second conjunct), and from Definition 29 (page 24) of $\mathbb{M}$ for 3-colored next functions (third conjunct).* □

Finally, given the previous two lemmas, we sketch a proof of Lemma 1 (page 27).

*Proof (Of Lemma 1).* Both I. and II. follow from induction on the length of the prefix of a painting. The base case (prefix of length 1) follows from the preservation of well-formedness of $\mathbb{M}$ for next functions and because $\lambda_0 \in \Lambda$ by Definition 31 (page 27). To prove the inductive step, first, suppose there exists a painting with a prefix of length $2j-1$ on which the lemma holds, for some $j \geq 1$ (note that the $(2j-1)$-th element is an index). Next, apply Lemma 2 to establish that there exists a painting with a prefix of length $2j$ on which the lemma holds (note that the $(2j)$-th element is a coloring). Finally, apply Lemma 3 to establish that there exists a painting with a prefix of length $2j+1 = 2(j+1)-1$ on which the lemma holds. □

**Simulation** Alternatively, we can define the correctness of $\mathbb{M}$ in terms of simulation. In this context, we call the $\mathbb{M}$-transformation correct if, for each 3-colored $\eta$-connector $\mathcal{C}^{\mathsf{Col}}$, its $\mathbb{M}$-transformation $\mathbb{M}(\mathcal{C}^{\mathsf{Col}})$ *simulates* $\mathcal{C}^{\mathsf{Col}}$. Shortly, we explain the meaning of simulation for $\eta$-connectors. First, however, we introduce an auxiliary concept: simulation for colorings. Informally, a coloring $c_1$ simulates a coloring $c_2$ if $c_1$ assigns the same color as $c_2$ to all the nodes in the domain of $c_2$. For simplicity of the presentation, we assume the no-flow color - - - - equal to both - -▷- - and - -◁- -.

**Definition 32 (Simulation for colorings).** *Let $c_1$ and $c_2$ be colorings over $N_1$ and $N_2$. $c_1$ simulates $c_2$, denoted $c_1 \succeq c_2$, if $N_2 \subseteq N_1$ and $c_1(n) = c_2(n)$ for all $n \in N_2$.*

Note that simulation relates asymmetrically: $c_1 \succeq c_2$ does not imply $c_2 \succeq c_1$. To see this, suppose $c_1 = \{ A \mapsto$ ——, $B \mapsto$ —— $\}$ and $c_2 = \{ A \mapsto$ —— $\}$: whereas $c_1 \succeq c_2$, $c_2 \not\succeq c_1$ (because the domain of $c_2$ does not include the domain of $c_1$). The following lemma states that the $\mathbb{M}$-transformation of a 3-coloring—i.e., a 2-coloring—simulates the original 3-coloring.

**Lemma 4 ($\mathbb{M}$-ed 3-colorings simulate 3-colorings).** *Let $c$ be a 3-coloring over $N$. Then, $\mathbb{M}(c) \succeq c$.*

*Proof. By Definition 26 (page 21) of $\mathbb{M}$ for 3-colorings, [$\{ n \mapsto$ ——, $\overline{n} \mapsto$ - - - - $\} \subseteq \mathbb{M}(c)$ iff $\{ n \mapsto$ —— $\} \subseteq c$] and [[$\{ n \mapsto$ - - - -, $\overline{n} \mapsto$ —— $\} \subseteq \mathbb{M}(c)$ or $\{ n \mapsto$ - - - -, $\overline{n} \mapsto$ - - - - $\} \subseteq \mathbb{M}(c)$] iff [$\{ n \mapsto$ - -▷- - $\} \subseteq c$ or $\{ n \mapsto$ - -◁- - $\} \subseteq c$]], for all $n \in N$. Because $c$ is a 3-coloring over $N$ by the premise, $\mathbb{M}(c) \succeq c$.* □

We proceed with the definition of simulation for $\eta$-connectors. Our definition roots in Milner's notion of *bi-simulation* [Mil89], adapted to fit coloring models. Informally, an $\eta$-connector $\mathcal{C}_1^{\mathsf{Col}}$ simulates an $\eta$-connector $\mathcal{C}_2^{\mathsf{Col}}$ if the former describes state-transitions that *correspond* to the state-transitions described by $\mathcal{C}_2^{\mathsf{Col}}$; we define correspondence of state-transitions in terms of simulation for colorings.

**Definition 33 (Simulation for $\eta$-connectors).** *Let $\mathcal{C}_1^{\mathsf{Col}} = \langle C_1, \eta_1 \rangle$ and $\mathcal{C}_2^{\mathsf{Col}} = \langle C_2, \eta_2 \rangle$ be $\eta$-connectors over $[N_1, \Lambda_1, S_1]$ and $[N_2, \Lambda_2, S_2]$. $\mathcal{C}_1^{\mathsf{Col}}$ simulates $\mathcal{C}_2^{\mathsf{Col}}$, denoted $\mathcal{C}_1^{\mathsf{Col}} \succeq \mathcal{C}_2^{\mathsf{Col}}$, if there exists a relation $\mathcal{R} \subseteq \Lambda_1 \times \Lambda_2$ such that, for all $\langle \lambda_1, \lambda_2 \rangle \in \mathcal{R}$:*

> *if $[\langle \lambda_2, c_2 \rangle \mapsto \lambda_2'] \in \eta_2$ then there exists a $\lambda_1' \in \Lambda_1$ such that $[\langle \lambda_1, c_1 \rangle \mapsto \lambda_1'] \in \eta_1$ and $\langle \lambda_1', \lambda_2' \rangle \in \mathcal{R}$ and $c_1 \succeq c_2$.*

*In that case, $\mathcal{R}$ is called a simulation relation.*

Like simulation for colorings, simulation for $\eta$-connectors relates asymmetrically (in contrast to Milner's bi-simulation). We remark that one can easily adapt the previous two definitions to satisfy symmetry—i.e., by replacing $\subseteq$ with $=$ in Definition 32 (page 29) and by supplementing the condition of $\mathcal{R}$ in Definition 33 (page 29) with a symmetric condition. As we do not need these definitions in this report, however, we skip them.

The following lemma states the correctness of $\mathbb{M}$ (in the context of simulation): it establishes that the application of $\mathbb{M}$ to a 3-colored $\eta$-connector $\mathcal{C}^{\mathsf{Col}}$ yields a 2-colored $\eta$-connector that simulates $\mathcal{C}^{\mathsf{Col}}$. This means that $\mathbb{M}$-ed 3-colored $\eta$-connectors behave indistinguishably from their untransformed counterparts if we disregard their context nodes.

**Lemma 5 ($\mathbb{M}$-ed 3-colored $\eta$-connectors simulate 3-colored $\eta$-connectors).** *Let $\mathcal{C}^{\mathsf{Col}} = \langle C, \eta \rangle$ be a 3-colored $\eta$-connector over $[N, \Lambda, S]$. Then, $\mathbb{M}(\mathcal{C}^{\mathsf{Col}}) \succeq \mathcal{C}^{\mathsf{Col}}$.*

*Proof.* *Let* $\mathcal{R} = \{ \langle \lambda, \lambda \rangle \mid \lambda \in \Lambda \}$. *We show that* $\mathcal{R}$ *is a simulation relation. Let* $\langle \lambda, \lambda \rangle \in \mathcal{R}$. *Suppose* $[\langle \lambda, c \rangle \mapsto \lambda'] \in \eta$. *First, by Definition 29 (page 24) of* $\mathbb{M}$ *for 3-colored next functions,* $[\langle \lambda, \mathbb{M}(c) \rangle \mapsto \lambda'] \in \mathbb{M}(\eta)$. *Next, by the definition of* $\mathcal{R}$, $\langle \lambda', \lambda' \rangle \in \mathcal{R}$. *Finally, because* $\mathbb{M}$*-ed 3-colorings simulate 3-colorings by Lemma 4 (page 29),* $\mathbb{M}(c) \succeq c$. *Hence,* $\mathcal{R}$ *is a simulation relation. Therefore,* $\mathbb{M}(\mathcal{C}^{\mathsf{Col}}) \succeq \mathcal{C}^{\mathsf{Col}}$. □

### 3.3 Distributivity of $\mathbb{M}$

In the previous subsection, we showed the correctness of $\mathbb{M}$. Though an essential result, it not yet suffices for the following reason: to properly compose a 2-colored $\eta$-connector from context-dependent constituents, we still must compose a 3-colored $\eta$-connector from 3-colored constituents first. Only thereafter, we can apply $\mathbb{M}$ to obtain the desired 2-colored $\eta$-connector. Instead, we prefer (i) to apply $\mathbb{M}$ only once to the 3-colored $\eta$-connectors that model the common primitive connectors and (ii) to construct more complex 2-colored $\eta$-connectors by composing these $\mathbb{M}$-transformations. We favor this approach because we speculate that an implementation of Reo that operates on 2-coloring models can compute connector composition more efficiently than an implementation that operates on 3-coloring models.[11] In this subsection, we develop the theory that accommodates this: we show that the $\mathbb{M}$-transformation *distributes over composition*. The idea appears quite simple: we wish to prove that it does not matter whether we first compose 3-colored $\eta$-connectors $\mathcal{C}_1^{\mathsf{Col}}$ and $\mathcal{C}_2^{\mathsf{Col}}$ and then apply $\mathbb{M}$ to the composition or first apply $\mathbb{M}$ to $\mathcal{C}_1^{\mathsf{Col}}$ and $\mathcal{C}_2^{\mathsf{Col}}$ and then compose the resulting 2-colored $\eta$-connectors. Although straightforward from a conceptual perspective, the details involve quite some work. As in Section 3.1 (page 16), we divide our presentation into two parts: we commence with the distributivity of $\mathbb{M}$ for the structural aspects of 3-colored $\eta$-connectors and conclude with the distributivity of $\mathbb{M}$ for their behavioral aspects.

**Structure** We start with three lemmas that concern the distributivity of $\mathbb{M}$ for the structural aspects of 3-colored $\eta$-connectors. The first lemma states that $\mathbb{M}$ for sets of nodes distributes over the common set operators for union, intersection, and complement. In our proofs, we apply the commutativity and distributivity properties of these operators.

**Lemma 6 ($\mathbb{M}$ for sets of nodes distributes over union, intersection, and complement).** *Let* $N_1, N_2 \subseteq \mathcal{N}\mathsf{ode}$ *be sets of nodes. Then:*

$$\begin{aligned} \mathbb{M}(N_1) \cup \mathbb{M}(N_2) &= \mathbb{M}(N_1 \cup N_2) \\ \mathbb{M}(N_1) \cap \mathbb{M}(N_2) &= \mathbb{M}(N_1 \cap N_2) \\ \mathbb{M}(N_1) \setminus \mathbb{M}(N_2) &= \mathbb{M}(N_1 \setminus N_2) \end{aligned}$$

*Proof.* *Follows from Table 3 (page 31), Table 4 (page 31), and Table 5 (page 32).* □

[11] We leave an actual (proof-of-concept) implementation for future work.

**Table 3.** Proof: $\mathbb{M}(N_1) \cup \mathbb{M}(N_2) = \mathbb{M}(N_1 \cup N_2)$

$\mathbb{M}(N_1) \cup \mathbb{M}(N_2)$
$=$ /* By Definition 22 (page 17) of $\mathbb{M}$ for sets of nodes */
$\bigcup_{n \in N_1} \{ n, \overline{n} \} \cup \bigcup_{n \in N_2} \{ n, \overline{n} \}$
$=$ /* By the definition of $\cup$ */
$\bigcup_{n \in N_1 \cup N_2} \{ n, \overline{n} \}$
$=$ /* By Definition 22 (page 17) of $\mathbb{M}$ for sets of nodes */
$\mathbb{M}(N_1 \cup N_2)$

**Table 4.** Proof: $\mathbb{M}(N_1) \cap \mathbb{M}(N_2) = \mathbb{M}(N_1 \cap N_2)$

$\mathbb{M}(N_1) \cap \mathbb{M}(N_2)$
$=$ /* By Definition 22 (page 17) of $\mathbb{M}$ for sets of nodes */
$\bigcup_{n_1 \in N_1} \{ n_1, \overline{n_1} \} \cap \bigcup_{n_2 \in N_2} \{ n_2, \overline{n_2} \}$
$=$ /* By the commutativity of $\cap$, and by the distributivity of $\cap$ over $\bigcup$ */
$\bigcup_{n_1 \in N_1} \bigcup_{n_2 \in N_2} (\{ n_1, \overline{n_1} \} \cap \{ n_2, \overline{n_2} \})$
$=$ /* Because, by 1-TO-1 in Definition 22 (page 17), $\{ n_1, \overline{n_1} \} \cap \{ n_2, \overline{n_2} \} = \emptyset$ if $n_1 \neq n_2$ and $\{ n_1, \overline{n_1} \} \cap \{ n_2, \overline{n_2} \} = \{ n_1, \overline{n_1} \} = \{ n_2, \overline{n_2} \}$ otherwise */
$\bigcup_{n_1 \in N_1 \cap N_2} \bigcup_{n_2 \in N_1 \cap N_2} \mathbb{M}(n_1)$
$=$ /* Because $n_2$ is a dead variable */
$\bigcup_{n_1 \in N_1 \cap N_2} \mathbb{M}(n_1)$
$=$ /* By Definition 22 (page 17) of $\mathbb{M}$ for sets of nodes */
$\mathbb{M}(N_1 \cup N_2)$

Similar to the previous lemma, the next lemma states that $\mathbb{M}$ for sets of primitives distributes over the common set operators for union, intersection, and complement. Because we use only the first of these three properties in the sequel, we skip the proofs of the latter two and leave these as conjectures.

**Lemma 7 ($\mathbb{M}$ for sets of primitives distributes over union, intersection, and complement).** *Let $E_1$ and $E_2$ be sets of primitives. Then:*

$$\begin{aligned} \mathbb{M}(E_1) \cup \mathbb{M}(E_2) &= \mathbb{M}(E_1 \cup E_2) \\ \mathbb{M}(E_1) \cap \mathbb{M}(E_2) &= \mathbb{M}(E_1 \cap E_2) \\ \mathbb{M}(E_1) \setminus \mathbb{M}(E_2) &= \mathbb{M}(E_1 \setminus E_2) \end{aligned}$$

*Proof. Follows from Table 6 (page 32).* □

The final lemma of this part states that $\mathbb{M}$ for connector structures distributes over composition. This means that it does not matter whether we first compose connector structures $C_1$ and $C_2$ and then apply $\mathbb{M}$ to the composition or first apply $\mathbb{M}$ to $C_1$ and $C_2$ and then compose the transformations; the resulting

**Table 5.** Proof: $\mathbb{M}(N_1) \setminus \mathbb{M}(N_2) = \mathbb{M}(N_1 \setminus N_2)$

| |
|---|
| $\mathbb{M}(N_1) \setminus \mathbb{M}(N_2)$ |
| $=$ /* By the definition of $\setminus$ */ |
| $\{\, n \in \mathbb{M}(N_1) \mid n \notin \mathbb{M}(N_2) \,\}$ |
| $=$ /* By the definition of $\bigcup$ */ |
| $\bigcup_{n \in \{n \in \mathbb{M}(N_1) \mid n \notin \mathbb{M}(N_2)\}} \{n\}$ |
| $=$ /* Because, by Definition 22 (page 17) of $\mathbb{M}$ for sets of nodes, $n \in \mathbb{M}(N)$ iff $\overline{n} \in \mathbb{M}(N)$ */ |
| $\bigcup_{n \in \{n \in N_1 \mid n \notin N_2\}} \{n, \overline{n}\}$ |
| $=$ /* By the definition of $\setminus$ */ |
| $\bigcup_{n \in N_1 \setminus N_2} \mathbb{M}(n)$ |
| $=$ /* By Definition 22 (page 17) of $\mathbb{M}$ for sets of nodes */ |
| $\mathbb{M}(N_1 \setminus N_2)$ |

**Table 6.** Proof: $\mathbb{M}(E_1) \cup \mathbb{M}(E_2) = \mathbb{M}(E_1 \cup E_2)$.

| |
|---|
| $\mathbb{M}(E_1) \cup \mathbb{M}(E_2)$ |
| $=$ /* By Definition 24 (page 18) of $\mathbb{M}$ for sets of primitives */ |
| $\{\, \mathbb{M}(e) \mid e \in E_1 \,\} \cup \{\, \mathbb{M}(e) \mid e \in E_2 \,\}$ |
| $=$ /* By the definition of $\cup$ */ |
| $\{\, \mathbb{M}(e) \mid e \in E_1 \cup E_2 \,\}$ |
| $=$ /* By Definition 24 (page 18) of $\mathbb{M}$ for sets of primitives */ |
| $\mathbb{M}(E_1 \cup E_2)$ |

connector structures equal each other. In our proof, we apply the distributivity lemmas that concern $\mathbb{M}$ for sets of nodes and $\mathbb{M}$ for sets of primitives.

**Lemma 8 ($\mathbb{M}$ for connector structures distributes over composition).** *Let $C_1 = \langle B_1, E_1 \rangle$ and $C_2 = \langle B_2, E_2 \rangle$ be connector structures over $N_1$ and $N_2$. Then:*

$$\mathbb{M}(C_1) \boxtimes \mathbb{M}(C_2) = \mathbb{M}(C_1 \boxtimes C_2)$$

*Proof. Applying Definition 25 (page 18) of $\mathbb{M}$ for connector structures and Definition 4 (page 4) of composition for connector structures yields:*

$$\left\langle \begin{array}{l} \mathbb{M}(B_1) \cup \mathbb{M}(B_2) \setminus \mathbb{M}(B_1) \cap \mathbb{M}(B_2), \\ \mathbb{M}(E_1) \cup \mathbb{M}(E_2) \end{array} \right\rangle = \left\langle \begin{array}{l} \mathbb{M}(B_1 \cup B_2 \setminus B_1 \cap B_2), \\ \mathbb{M}(E_1 \cup E_2) \end{array} \right\rangle \begin{array}{l} \text{(I)} \\ \text{(II)} \end{array}$$

*Sub-equation* (I) *follows from the distributivity of $\mathbb{M}$ for sets of nodes over union, intersection, and complement in Lemma 6 (page 30), while sub-equation* (II) *follows from the distributivity of $\mathbb{M}$ for sets of primitives over union in Lemma 7 (page 31). Therefore, $\mathbb{M}(C_1) \boxtimes \mathbb{M}(C_2) = \mathbb{M}(C_1 \boxtimes C_2)$.* □

**Table 7.** Proof: $\mathbb{M}(c_1) \cup \mathbb{M}(c_2) = \mathbb{M}(c_1 \cup c_2)$.

$\mathbb{M}(\mathcal{C}_1^{\mathsf{Col}}) \cup \mathbb{M}(\mathcal{C}_2^{\mathsf{Col}})$

$= $ /* By Definition 12 (page 10) of composition for colorings and because, since $\mathbb{M}$-ed 3-colorings are 2-colorings by Proposition 5 (page 21), $\mathbb{M}(\mathcal{C}_1^{\mathsf{Col}})$ and $\mathbb{M}(\mathcal{C}_2^{\mathsf{Col}})$ are 2-colorings over $\mathbb{M}(N_1)$ and $\mathbb{M}(N_2)$ */

$$\left\{ n \mapsto \kappa \;\middle|\; n \in \mathbb{M}(N_1) \cup \mathbb{M}(N_2) \text{ and } \kappa = \begin{pmatrix} (\mathbb{M}(c_1))(n) \text{ if } n \in \mathbb{M}(N_1) \\ (\mathbb{M}(c_2))(n) \text{ otherwise} \end{pmatrix} \right\}$$

$= $ /* By the definition of $\bigcup$ */

$$\bigcup_{n \in \mathbb{M}(N_1) \cup \mathbb{M}(N_2)} \begin{cases} \{ n \mapsto (\mathbb{M}(c_1))(n) \} \text{ if } n \in \mathbb{M}(N_1) \\ \{ n \mapsto (\mathbb{M}(c_2))(n) \} \text{ otherwise} \end{cases}$$

$= $ /* By the distributivity of $\mathbb{M}$ for sets of nodes over $\cup$ in Lemma 6 (page 30) */

$$\bigcup_{n \in \mathbb{M}(N_1 \cup N_2)} \begin{cases} \{ n \mapsto (\mathbb{M}(c_1))(n) \} \text{ if } n \in \mathbb{M}(N_1) \\ \{ n \mapsto (\mathbb{M}(c_2))(n) \} \text{ otherwise} \end{cases}$$

$= $ /* Because, by Definition 22 (page 17) of $\mathbb{M}$ for sets of nodes, $n \in \mathbb{M}(N_1 \cup N_2)$ iff $\overline{n} \in \mathbb{M}(N_1 \cup N_2)$ */

$$\bigcup_{n \in N_1 \cup N_2} \begin{cases} \{ n \mapsto (\mathbb{M}(c_1))(n), \overline{n} \mapsto (\mathbb{M}(c_1))(\overline{n}) \} \text{ if } n \in N_1 \\ \{ n \mapsto (\mathbb{M}(c_2))(n), \overline{n} \mapsto (\mathbb{M}(c_2))(\overline{n}) \} \text{ otherwise} \end{cases}$$

$= $ /* By Definition 26 (page 21) of $\mathbb{M}$ for 3-colorings */

$$\bigcup_{n \in N_1 \cup N_2} \begin{cases} \mathbb{M}(\{ n \mapsto c_1(n) \}) \text{ if } n \in N_1 \\ \mathbb{M}(\{ n \mapsto c_2(n) \}) \text{ otherwise} \end{cases}$$

$= $ /* By Definition 12 (page 10) of composition for colorings */

$$\bigcup_{n \in N_1 \cup N_2} \mathbb{M}(n \mapsto (c_1 \cup c_2)(n))$$

$= $ /* By Definition 26 (page 21) of $\mathbb{M}$ for 3-colorings */

$\mathbb{M}(c_1 \cup c_2)$

**Behavior** We proceed with five lemmas that concern the distributivity of $\mathbb{M}$ for the behavioral aspects of 3-colored $\eta$-connectors.

The first lemma states that $\mathbb{M}$ for 3-coloring tables distributes over composition. This means that it does not matter whether we first compose 3-colorings $c_1$ and $c_2$ and then apply $\mathbb{M}$ to the composition or first apply $\mathbb{M}$ to $c_1$ and $c_2$ and then compose the transformations; the resulting 2-colorings equal each other. In our proof, we apply the one-to-one correspondence between base nodes and context nodes.

**Lemma 9 ($\mathbb{M}$ for 3-colorings distributes over composition).** *Let $c_1$ and $c_2$ be 3-colorings over $N_1$ and $N_2$ such that $c_1(n) = c_2(n)$ for all $n \in N_1 \cap N_2$. Then:*

$$\mathbb{M}(c_1) \cup \mathbb{M}(c_2) = \mathbb{M}(c_1 \cup c_2)$$

*Proof. Follows from Table 7 (page 33).* □

The second lemma states that $\mathbb{M}$ for 3-coloring tables distributes over composition. This means that it does not matter whether we first compose 3-coloring tables $T_1$ and $T_2$ and then apply $\mathbb{M}$ to the composition or first apply $\mathbb{M}$ to $T_1$ and

**Table 8.** Proof: $\varphi(\mathbb{M}(c_1), \mathbb{M}(c_2)) \equiv \varphi(c_1, c_2)$

| |
|---|
| $\varphi(\mathbb{M}(c_1), \mathbb{M}(c_2))$ |
| $\equiv$ /* By the definition of $\varphi$ */ |
| $(\mathbb{M}(c_1))(n) = (\mathbb{M}(c_2))(n)$ for all $n \in \mathbb{M}(N_1) \cap \mathbb{M}(N_2)$ |
| $\equiv$ /* By the distributivity of $\mathbb{M}$ for sets of nodes over union in Lemma 6 (page 30) */ |
| $(\mathbb{M}(c_1))(n) = (\mathbb{M}(c_2))(n)$ for all $n \in \mathbb{M}(N_1 \cap N_2)$ |
| $\equiv$ /* Because, by Definition 22 (page 17) of $\mathbb{M}$ for sets of nodes, $n \in \mathbb{M}(N_1 \cap N_2)$ iff $\overline{n} \in \mathbb{M}(N_1 \cap N_2)$ */ |
| $(\mathbb{M}(c_1))(n) = (\mathbb{M}(c_2))(n)$ and $(\mathbb{M}(c_1))(\overline{n}) = (\mathbb{M}(c_2))(\overline{n})$ for all $n \in N_1 \cap N_2$ |
| $\equiv$ /* By Definition 26 (page 21) of $\mathbb{M}$ for 3-colorings */ |
| $c_1(n) = c_2(n)$ for all $n \in N_1 \cap N_2$ |
| $\equiv$ /* By the definition of $\varphi$ */ |
| $\varphi(c_1, c_2)$ |

$T_2$ and then compose the transformations; the resulting 2-coloring tables equal each other. Our proof turns out less straightforwardly than one might expect, because we must ensure that the compatibility condition—i.e., colorings must assign the same colors to their shared nodes—holds. This requires an auxiliary derivation in which we show that the compatibility of 3-colorings $c_1$ and $c_2$ implies the compatibility of their $\mathbb{M}$-transformations and vice versa. In our main proof, in addition to this result, we apply the distributivity lemma that concerns $\mathbb{M}$ for 3-colorings.

**Lemma 10 ($\mathbb{M}$ for 3-coloring tables distributes over composition).** *Let $T_1$ and $T_2$ be a 3-coloring tables over $N_1$ and $N_2$. Then:*

$$\mathbb{M}(T_1) \cdot \mathbb{M}(T_2) = \mathbb{M}(T_1 \cdot T_2)$$

*Proof. First, we prove an auxiliary equivalence that helps us keep the main proof more concise. Let $\varphi(c, c')$ be a predicate on colorings $c$ and $c'$ over $N$ and $N'$ defined as $[\varphi(c, c) \equiv c(n) = c'(n)$ for all $n \in N \cap N']$. It follows from Table 8 (page 34) that $\varphi(\mathbb{M}(c_1), \mathbb{M}(c_2)) \equiv \varphi(c_1, c_2)$. Then, the lemma follows from Table 9 (page 35).* □

The third lemma states that $\mathbb{M}$ for 3-CTMs distributes over composition. This means that it does not matter whether we first compose 3-CTMs $S_1$ and $S_2$ and then apply $\mathbb{M}$ to the composition or first apply $\mathbb{M}$ to $S_1$ and $S_2$ and then compose the transformations; the resulting 2-CTMs equal each other. In our proof, we apply the distributivity lemma that concerns $\mathbb{M}$ for 3-coloring tables.

**Lemma 11 ($\mathbb{M}$ for 3-CTMs distributes over composition).** *Let $S_1$ and $S_2$ be CTMs over $[N_1, \Lambda_1]$ and $[N_2, \Lambda_2]$. Then:*

$$\mathbb{M}(S_1) \odot \mathbb{M}(S_2) = \mathbb{M}(S_1 \odot S_2)$$

**Table 9.** Proof: $\mathbb{M}(T_1) \cdot \mathbb{M}(T_2) = \mathbb{M}(T_1 \cdot T_2)$

$\mathbb{M}(T_1) \cdot \mathbb{M}(T_2)$
$= $ /* By Definition 13 (page 10) of composition for coloring tables */
$\{\, \mathbb{M}(c_1) \cup \mathbb{M}(c_2) \mid \mathbb{M}(c_1) \in \mathbb{M}(T_1)$ and $\mathbb{M}(c_2) \in \mathbb{M}(T_2)$ and $\varphi(\mathbb{M}(c_1), \mathbb{M}(c_2)) \,\}$
$= $ /* By the distributivity of $\mathbb{M}$ for 3-colorings over composition in Lemma 9 (page 33) and because, by Definition 27 (page 22) of $\mathbb{M}$ for 3-coloring tables, $[\mathbb{M}(c_1) \in \mathbb{M}(T_1)$ iff $c_1 \in T_1]$ and $[\mathbb{M}(c_2) \in \mathbb{M}(T_2)$ iff $c_2 \in T_2]$ */
$\{\, \mathbb{M}(c_1 \cup c_2) \mid c_1 \in T_1$ and $c_2 \in T_2$ and $\varphi(\mathbb{M}(c_1), \mathbb{M}(c_2)) \,\}$
$= $ /* By Table 8 (page 34) */
$\{\, \mathbb{M}(c_1 \cup c_2) \mid c_1 \in T_1$ and $c_2 \in T_2$ and $\varphi(c_1, c_2) \,\}$
/* Because $[c_1 \in T_1$ and $c_2 \in T_2$ and $\varphi(c_1, c_2)]$ iff $[c_1 \cup c_2 \in \{\, c_1 \cup c_2 \mid c_1 \in T_1$ and $c_2 \in T_2$ and $\varphi(c_1, c_2) \,\}]$ */
$\{\, \mathbb{M}(c_1 \cup c_2) \mid c_1 \cup c_2 \in \{\, c_1 \cup c_2 \mid c_1 \in T_1$ and $c_2 \in T_2$ and $\varphi(c_1, c_2) \,\} \,\}$
$= $ /* By Definition 13 (page 10) of composition for coloring tables */
$\{\, \mathbb{M}(c_1 \cup c_2) \mid c_1 \cup c_2 \in T_1 \cdot T_2 \,\}$
$= $ /* By Definition 27 (page 22) of $\mathbb{M}$ for 3-coloring tables */
$\mathbb{M}(T_1 \cdot T_2)$

*Proof. Follows from Table 10 (page 36).* □

The fourth lemma states that $\mathbb{M}$ for 3-colored next functions distributes over composition. This means that it does not matter whether we first compose 3-colored next functions $\eta_1$ and $\eta_2$ and then apply $\mathbb{M}$ to the composition or first apply $\mathbb{M}$ to $\eta_1$ and $\eta_2$ and then compose the transformations; the resulting 2-colored next functions equal each other. In our proof, we apply the distributivity lemmas that concern $\mathbb{M}$ for 3-colorings and $\mathbb{M}$ for 3-CTMs.

**Lemma 12 ($\mathbb{M}$ for 3-colored next functions distributes over composition).** *Let $\eta_1$ and $\eta_2$ be next functions over $[N_1, \Lambda_1, S_1]$ and $[N_2, \Lambda_2, S_2]$. Then:*

$$\mathbb{M}(\eta_1) \otimes \mathbb{M}(\eta_2) = \mathbb{M}(\eta_1 \otimes \eta_2)$$

*Proof. Follows from Table 11 (page 37)* □

The fifth and final lemma states that $\mathbb{M}$ for 3-colored $\eta$-connectors distributes over composition. This means that it does not matter whether we first compose 3-colored $\eta$-connectors $\mathcal{C}_1^{\mathsf{Col}}$ and $\mathcal{C}_2^{\mathsf{Col}}$ and then apply $\mathbb{M}$ to the composition or first apply $\mathbb{M}$ to $\mathcal{C}_1^{\mathsf{Col}}$ and $\mathcal{C}_2^{\mathsf{Col}}$ and then compose the transformations; the resulting 2-colored $\eta$-connectors equal each other. In our proof, we apply the distributivity lemmas that concern $\mathbb{M}$ for connector structures and $\mathbb{M}$ for 3-colored next functions.

**Lemma 13 ($\mathbb{M}$ for 3-colored $\eta$-connectors distributes over composition).** *Let $\mathcal{C}_1^{\mathsf{Col}} = \langle C_1, \eta_1 \rangle$ and $\mathcal{C}_2^{\mathsf{Col}} = \langle C_2, \eta_2 \rangle$ be 3-colored $\eta$-connectors over*

**Table 10.** Proof: $\mathbb{M}(S_1) \odot \mathbb{M}(S_2) = \mathbb{M}(S_1 \odot S_2)$

$$
\begin{array}{ll}
 & \mathbb{M}(S_1) \odot \mathbb{M}(S_2) \\
= & /^* \text{ By Definition 28 (page 23) of } \mathbb{M} \text{ for 3-CTMs } {}_*/ \\
 & \{\, \lambda_1 \mapsto \mathbb{M}(S_1(\lambda_1)) \mid \lambda_1 \in \Lambda_1 \,\} \odot \{\, \lambda_2 \mapsto \mathbb{M}(S_1(\lambda_2)) \mid \lambda_1 \in \Lambda_2 \,\} \\
= & /^* \text{ By Definition 14 (page 11) of composition for CTMs } {}_*/ \\
 & \{\, \langle \lambda_1, \lambda_2 \rangle \mapsto \mathbb{M}(S_1(\lambda_1)) \cdot \mathbb{M}(S_2(\lambda_2)) \mid \lambda_1 \in \Lambda_1 \text{ and } \lambda_2 \in \Lambda_2 \,\} \\
= & /^* \text{ By the distributivity of } \mathbb{M} \text{ for 3-coloring tables over composition in Lemma 10 (page 34) } {}_*/ \\
 & \{\, \langle \lambda_1, \lambda_2 \rangle \mapsto \mathbb{M}(S_1(\lambda_1) \cdot S_2(\lambda_2)) \mid \lambda_1 \in \Lambda_1 \text{ and } \lambda_2 \in \Lambda_2 \,\} \\
= & /^* \text{ Because, by the definition of the Cartesian product, } [\lambda_1 \in \Lambda_1 \text{ and } \lambda_2 \in \Lambda_2] \text{ iff } \langle \lambda_1, \lambda_2 \rangle \in \Lambda_1 \times \Lambda_2 \; {}_*/ \\
 & \{\, \langle \lambda_1, \lambda_2 \rangle \mapsto \mathbb{M}(S_1(\lambda_1) \cdot S_2(\lambda_2)) \mid \langle \lambda_1, \lambda_2 \rangle \in \Lambda_1 \times \Lambda_2 \,\} \\
= & /^* \text{ By Definition 28 (page 23) of } \mathbb{M} \text{ for 3-CTMs } {}_*/ \\
 & \mathbb{M}(\{\, \langle \lambda_1, \lambda_2 \rangle \mapsto S_1(\lambda_1) \cdot S_2(\lambda_2) \mid \langle \lambda_1, \lambda_2 \rangle \in \Lambda_1 \times \Lambda_2 \,\}) \\
= & /^* \text{ By Definition 14 (page 11) of composition for CTMs } {}_*/ \\
 & \mathbb{M}(S_1 \odot S_2)
\end{array}
$$

$[N_1, \Lambda_1, S_1]$ *and* $[N_2, \Lambda_2, S_2]$. *Then:*

$$\mathbb{M}(\mathcal{C}_1^{\mathsf{Col}}) \times \mathbb{M}(\mathcal{C}_2^{\mathsf{Col}}) = \mathbb{M}(\mathcal{C}_1^{\mathsf{Col}} \times \mathcal{C}_2^{\mathsf{Col}})$$

*Proof. Applying Definition 30 (page 25) of* $\mathbb{M}$ *for 3-colored* $\eta$*-connectors and Definition 16 (page 12) of composition for* $\eta$*-connectors yields:*

$$\left\langle \begin{array}{l} \mathbb{M}(C_1) \boxtimes \mathbb{M}(C_2), \\ \mathbb{M}(\eta_1) \otimes \mathbb{M}(\eta_2) \end{array} \right\rangle = \left\langle \begin{array}{ll} \mathbb{M}(C_1 \boxtimes C_2), & \text{(I)} \\ \mathbb{M}(\eta_1 \otimes \eta_2) & \text{(II)} \end{array} \right.$$

*Sub-equation* (I) *follows from the distributivity of* $\mathbb{M}$ *for connector structures over composition in Lemma 8 (page 32), while sub-equation* (II) *follows from the distributivity of* $\mathbb{M}$ *for 3-colored next functions over composition in Lemma 12 (page 35). Therefore,* $\mathbb{M}(\mathcal{C}_1^{\mathsf{Col}}) \times \mathbb{M}(\mathcal{C}_2^{\mathsf{Col}}) = \mathbb{M}(\mathcal{C}_1^{\mathsf{Col}} \times \mathcal{C}_2^{\mathsf{Col}})$. □

## 3.4 Inverse of $\mathbb{M}$

Before we continue, let us recapitulate what we presented so far. First, in Section 3.1 (page 16), we defined the $\mathbb{M}$ operator for various types of operands. Next, in Section 3.2 (page 27), we proved the correctness of $\mathbb{M}$, and in Section 3.3 (page 30), we showed that $\mathbb{M}$ distributes over composition. In this last subsection, we investigate the *inverse* of $\mathbb{M}$. Essentially, we answer the following question, relevant to the computation of *connector animations*: can we transform $\mathbb{M}$-ed 3-colored $\eta$-connectors back to their original form?

**Table 11.** Proof: $\mathbb{M}(\eta_1) \otimes \mathbb{M}(\eta_2) = \mathbb{M}(\eta_1 \otimes \eta_2)$

$$
\begin{aligned}
& \mathbb{M}(\eta_1) \otimes \mathbb{M}(\eta_2) \\
= \; & /^* \text{ By Definition 29 (page 24) of } \mathbb{M} \text{ for 3-colored next functions } {}^*/ \\
& \{\, \lambda_1, \mathbb{M}(c_1) \mapsto \eta_1(\lambda_1, c_1)) \mid \lambda_1 \in \Lambda_1 \text{ and } c_1 \in S_1(\lambda_1) \,\} \\
& \qquad\qquad \otimes \\
& \{\, \lambda_2, \mathbb{M}(c_2) \mapsto \eta_2(\lambda_2, c_2)) \mid \lambda_2 \in \Lambda_2 \text{ and } c_2 \in S_2(\lambda_2) \,\} \\
= \; & /^* \text{ By Definition 15 of composition for next functions } {}^*/ \\
& \left\{ \begin{array}{c|c} \langle \lambda_1, \lambda_2 \rangle, \mathbb{M}(c_1) \cup \mathbb{M}(c_2) & \langle \lambda_1, \lambda_2 \rangle \in \Lambda_1 \times \Lambda_2 \\ \downarrow & \text{and} \\ \langle \eta_1(\lambda_1, c_1), \eta_2(\lambda_2, c_2) \rangle & \mathbb{M}(c_1) \cup \mathbb{M}(c_2) \in (\mathbb{M}(S_1) \odot \mathbb{M}(S_2))(\langle \lambda_1, \lambda_2 \rangle) \end{array} \right\} \\
= \; & /^* \text{ By the distributivity of } \mathbb{M} \text{ for 3-colorings over composition in Lemma 9 (page 33) } {}^*/ \\
& \left\{ \begin{array}{c|c} \langle \lambda_1, \lambda_2 \rangle, \mathbb{M}(c_1 \cup c_2) & \langle \lambda_1, \lambda_2 \rangle \in \Lambda_1 \times \Lambda_2 \\ \downarrow & \text{and} \\ \langle \eta_1(\lambda_1, c_1), \eta_2(\lambda_2, c_2) \rangle & \mathbb{M}(c_1 \cup c_2) \in (\mathbb{M}(S_1) \odot \mathbb{M}(S_2))(\langle \lambda_1, \lambda_2 \rangle) \end{array} \right\} \\
= \; & /^* \text{ By the distributivity of } \mathbb{M} \text{ for 3-CTMs over composition in Lemma 11 (page 34) } {}^*/ \\
& \left\{ \begin{array}{c|c} \langle \lambda_1, \lambda_2 \rangle, \mathbb{M}(c_1 \cup c_2) & \langle \lambda_1, \lambda_2 \rangle \in \Lambda_1 \times \Lambda_2 \\ \downarrow & \text{and} \\ \langle \eta_1(\lambda_1, c_1), \eta_2(\lambda_2, c_2) \rangle & \mathbb{M}(c_1 \cup c_2) \in (\mathbb{M}(S_1 \odot S_2))(\langle \lambda_1, \lambda_2 \rangle) \end{array} \right\} \\
= \; & /^* \text{ By Definition 28 (page 23) of } \mathbb{M} \text{ for 3-CTMs } {}^*/ \\
& \left\{ \begin{array}{c|c} \langle \lambda_1, \lambda_2 \rangle, \mathbb{M}(c_1 \cup c_2) & \langle \lambda_1, \lambda_2 \rangle \in \Lambda_1 \times \Lambda_2 \\ \downarrow & \text{and} \\ \langle \eta_1(\lambda_1, c_1), \eta_2(\lambda_2, c_2) \rangle & \mathbb{M}(c_1 \cup c_2) \in \mathbb{M}((S_1 \odot S_2)(\langle \lambda_1, \lambda_2 \rangle)) \end{array} \right\} \\
= \; & /^* \text{ By Definition 27 (page 22) of } \mathbb{M} \text{ for 3-coloring tables } {}^*/ \\
& \left\{ \begin{array}{c|c} \langle \lambda_1, \lambda_2 \rangle, \mathbb{M}(c_1 \cup c_2) & \langle \lambda_1, \lambda_2 \rangle \in \Lambda_1 \times \Lambda_2 \\ \downarrow & \text{and} \\ \langle \eta_1(\lambda_1, c_1), \eta_2(\lambda_2, c_2) \rangle & c_1 \cup c_2 \in (S_1 \odot S_2)(\langle \lambda_1, \lambda_2 \rangle) \end{array} \right\} \\
= \; & /^* \text{ By Definition 29 (page 24) of } \mathbb{M} \text{ for 3-colored next functions } {}^*/ \\
& \mathbb{M} \left( \left\{ \begin{array}{c|c} \langle \lambda_1, \lambda_2 \rangle, c_1 \cup c_2 & \langle \lambda_1, \lambda_2 \rangle \in \Lambda_1 \times \Lambda_2 \\ \downarrow & \text{and} \\ \langle \eta_1(\lambda_1, c_1), \eta_2(\lambda_2, c_2) \rangle & c_1 \cup c_2 \in (S_1 \odot S_2)(\langle \lambda_1, \lambda_2 \rangle) \end{array} \right\} \right) \\
= \; & /^* \text{ By Definition 15 (page 12) of composition for next functions } {}^*/ \\
& \mathbb{M}(\eta_1 \otimes \eta_2)
\end{aligned}
$$

Connector animating [Cos10] concerns the visual simulation of Reo connectors; the current Reo distribution[12] includes a tool that facilitates this. To compute animations, this tool analyzes the coloring models of connectors. Experience suggests that animations based on 3-coloring models provide users with better insight about the behavior of a connector than animations based on 2-coloring models. To accord with this observation, however, becomes problematic if—for the sake of improving efficiency as speculated in the previous subsection—we

[12] Homepage: http://reo.project.cwi.nl/

**Table 12.** Proof: $\frac{1}{\mathbb{M}}(\mathbb{M}(N)) = N$.

| |
|---|
| $\frac{1}{\mathbb{M}}(\mathbb{M}(N))$ |
| $=$ /* By Definition 22 (page 17) of $\mathbb{M}$ */ |
| $\frac{1}{\mathbb{M}}\left(\bigcup_{n\in N}\{\, n, \overline{n}\,\}\right)$ |
| $=$ /* By Definition 34 (page 38) of $\frac{1}{\mathbb{M}}$ */ |
| $\{n' \in \bigcup_{n\in N}\{\, n, \overline{n}\,\} \mid n', \overline{n'} \in \bigcup_{n\in N}\{\, n, \overline{n}\,\}\}$ |
| $=$ /* Because, by 1-TO-1 in Definition 22 (page 17), $[n', \overline{n'} \in \bigcup_{n\in N}\{\, n, \overline{n}\,\}]$ iff $[n' \in \bigcup_{n\in N}\{\, n\,\}]$ and because, by the definition of $\bigcup$, $[n' \in \bigcup_{n\in N}\{\, n\,\}]$ iff $[n' \in N]$ */ |
| $\{n' \in \bigcup_{n\in N}\{\, n, \overline{n}\,\} \mid n' \in N\}$ |
| $=$ /* Because $\bigcup$ ranges over $N$ */ |
| $N$ |

would write an implementation of Reo that operates on $\mathbb{M}$-ed 3-coloring models instead of the 3-coloring models themselves: in that case, we would have to compute animations based on 2-coloring models, which inform users poorly. Therefore, in this final subsection, we define the inverse of $\mathbb{M}$, denoted by $\frac{1}{\mathbb{M}}$, which allows us to retrieve a 3-colored $\eta$-connector from its $\mathbb{M}$-transformation. We commence with the definitions of $\frac{1}{\mathbb{M}}$ for the structural aspects of coloring models and conclude with the definitions of $\frac{1}{\mathbb{M}}$ for their behavioral aspects.

**Structure** We start with $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed sets of nodes. This operator erases all the context nodes from the $\mathbb{M}$-ed set.

**Definition 34 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed sets of nodes).** *Let $N \subseteq \mathcal{N}\mathsf{ode}$ be an $\mathbb{M}$-ed set of nodes. Its $\frac{1}{\mathbb{M}}$-transformation, denoted $\frac{1}{\mathbb{M}}(N)$, is defined as:*

$$\tfrac{1}{\mathbb{M}}(N) = \{\, n \in N \mid n, \overline{n} \in N\,\}$$

The following lemma states that the application of $\frac{1}{\mathbb{M}}$ to an $\mathbb{M}$-ed set of nodes yields the untransformed set of nodes. In our proof, we apply the one-to-one correspondence between base nodes and context nodes.

**Lemma 14 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed sets of nodes inverts $\mathbb{M}$).** *Let $N \subseteq \mathcal{N}\mathsf{ode}$ be a set of nodes. Then:*

$$\tfrac{1}{\mathbb{M}}(\mathbb{M}(N)) = N$$

*Proof. Follows from Table 12 (page 38).* □

We proceed with the definition of $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed primitives. Similar to how $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed sets of nodes erases all the context nodes from a set, $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed primitives erases all the context nodes from a list.

**Definition 35 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed primitives).** *Let $e = (n_1^{j_1}, \ldots, n_{2k}^{j_{2k}})$ be an $\mathbb{M}$-ed primitive over $N$. Its $\frac{1}{\mathbb{M}}$-transformation, denoted $\frac{1}{\mathbb{M}}(e)$, is defined as:*

**Table 13.** Proof: $\frac{1}{\mathbb{M}}(\mathbb{M}(e)) = e$.

| |
|---|
| $\frac{1}{\mathbb{M}}(\mathbb{M}(e))$ |
| $=$ /* By Definition 2 (page 3) of $e$ */ |
| $\frac{1}{\mathbb{M}}(\mathbb{M}((n_1^{j_1}, \ldots, n_k^{j_k})))$ |
| $=$ /* By Definition 23 (page 17) of $\mathbb{M}$ for primitives */ |
| $\frac{1}{\mathbb{M}}((n_1^{j_1}, \ldots, n_k^{j_k}, (\overline{n_1})_{k+1}^{\neg j_1}, \ldots, (\overline{n_k})_{2k}^{\neg j_k}))$ |
| $=$ /* By Definition 35 (page 38) of $\frac{1}{\mathbb{M}}$ for primitives */ |
| $(n_1^{j_1}, \ldots, n_k^{j_k})$ |
| $=$ /* By Definition 2 (page 3) of $e$ */ |
| $e$ |

$$\frac{1}{\mathbb{M}}(e) = (n_1^{j_1}, \ldots, n_k^{j_k})$$

The following lemma states that the application of $\frac{1}{\mathbb{M}}$ to an $\mathbb{M}$-ed primitive yields the untransformed primitive.

**Lemma 15 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed primitives inverts $\mathbb{M}$).** *Let $e = (n_1^{j_1}, \ldots, n_k^{j_k})$ be a primitive over $N$. Then:*

$$\frac{1}{\mathbb{M}}(\mathbb{M}(e)) = e$$

*Proof. Follows from Table 13 (page 39).* □

We continue with the definition of $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed sets of primitives. Informally, the application of $\frac{1}{\mathbb{M}}$ to an $\mathbb{M}$-ed set of primitives yields a set of $\frac{1}{\mathbb{M}}$-ed $\mathbb{M}$-ed primitives.

**Definition 36 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed sets of primitives).** *Let $E$ be an $\mathbb{M}$-ed set of primitives. Its $\frac{1}{\mathbb{M}}$-transformation, denoted $\frac{1}{\mathbb{M}}(E)$, is defined as:*

$$\frac{1}{\mathbb{M}}(E) = \{\, \frac{1}{\mathbb{M}}(e) \mid e \in E \,\}$$

The following lemma states that the application of $\frac{1}{\mathbb{M}}$ to an $\mathbb{M}$-ed set of primitives yields the untransformed set of primitives. In our proof, we apply the inversion lemma that concerns $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed primitives.

**Lemma 16 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed sets of primitives inverts $\mathbb{M}$).** *Let $E$ be a set of primitives. Then:*

$$\frac{1}{\mathbb{M}}(\mathbb{M}(E)) = E$$

*Proof. Follows from Table 14 (page 40).* □

We end with $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed connector structures. Recall from Definition 3 (page 3) that connector structures consist of a set of boundary nodes and a set of primitives. This allows us to define $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed connector structures in terms of $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed sets of nodes and $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed sets of primitives.

**Table 14.** Proof: $\frac{1}{\mathbb{M}}(\mathbb{M}(E)) = E$

| |
|---|
| $\frac{1}{\mathbb{M}}(\mathbb{M}(E))$ |
| $=$ /* By Definition 24 (page 18) of $\mathbb{M}$ for sets of primitives */ |
| $\frac{1}{\mathbb{M}}(\{\,\mathbb{M}(e)\,\mid e \in E\,\})$ |
| $=$ /* By Definition 36 (page 39) of $\frac{1}{\mathbb{M}}$ for sets of primitives */ |
| $\{\,\frac{1}{\mathbb{M}}(\mathbb{M}(e))\,\mid \mathbb{M}(e) \in \{\,\mathbb{M}(e)\,\mid e \in E\,\}\,\}$ |
| /* By the inversion of $\mathbb{M}$ for primitives by $\frac{1}{\mathbb{M}}$ in Lemma 15 (page 39) and because $[\mathbb{M}(e) \in \{\,\mathbb{M}(e)\,\mid e \in E\,\}]$ iff $[e \in E]$ */ |
| $= \{\,e\,\mid e \in E\,\}$ |
| /* Because $E = \{\,e\,\mid e \in E\,\}$ */ |
| $= E$ |

**Definition 37 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed connector structures).** *Let $C = \langle B, E\rangle$ be an $\mathbb{M}$-ed connector structure over $N$. Its $\frac{1}{\mathbb{M}}$-transformation, denoted $\frac{1}{\mathbb{M}}(C)$, is defined as:*

$$\tfrac{1}{\mathbb{M}}(C) = \langle \tfrac{1}{\mathbb{M}}(B), \tfrac{1}{\mathbb{M}}(E)\rangle$$

The following lemma states that the application of $\frac{1}{\mathbb{M}}$ to an $\mathbb{M}$-ed connector structure yields the untransformed connector structure. In our proof, we apply the inversion lemmas that concern $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed sets of nodes and $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed sets of primitives.

**Lemma 17 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed connector structures inverts $\mathbb{M}$).** *Let $C = \langle N, B, E\rangle$ be a connector structure over $N$. Then:*

$$\tfrac{1}{\mathbb{M}}(\mathbb{M}(C)) = C$$

*Proof. Applying Definition 30 (page 25) of $\mathbb{M}$ for connector structures and Definition 42 (page 43) of $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed connector structures yields:*

$$\left\langle \begin{matrix} \frac{1}{\mathbb{M}}(\mathbb{M}(B)), \\ \frac{1}{\mathbb{M}}(\mathbb{M}(E)) \end{matrix} \right\rangle = \left\langle \begin{matrix} B, \\ E \end{matrix} \right\rangle \begin{matrix} \text{(I)} \\ \text{(II)} \end{matrix}$$

*Sub-equation* (II) *follows from the inversion of $\mathbb{M}$ for sets of nodes by $\frac{1}{\mathbb{M}}$ in Lemma 14 (page 38), while sub-equation* (II) *follows from the inversion of $\mathbb{M}$ for sets of primitives by $\frac{1}{\mathbb{M}}$ in Lemma 16 (page 39). Therefore, $\frac{1}{\mathbb{M}}(\mathbb{M}(C)) = C$.* □

**Behavior** We conclude with $\frac{1}{\mathbb{M}}$ for the behavioral aspects of $\mathbb{M}$-ed 3-colored $\eta$-connectors and begin with the definition of $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colorings. By analyzing the colors that an $\mathbb{M}$-ed 3-coloring assigns to a base node and its dual, we can infer the color that the untransformed 3-coloring assigns to the base node: due to Definition 26 (page 21) of $\mathbb{M}$ for 3-colorings, the colors (from 2-$\mathcal{C}$olor) that an $\mathbb{M}$-ed 3-coloring assigns to a base node and its dual uniquely define the color (from 3-$\mathcal{C}$olor) that the untransformed 3-coloring assigns to this base node.

**Table 15.** Proof: $\frac{1}{\mathbb{M}}(\mathbb{M}(c)) = c$

$\frac{1}{\mathbb{M}}(\mathbb{M}(c))$

$= $ /* By Definition 38 (page 41) of $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colorings */

$$\left\{ n \mapsto \kappa \;\middle|\; \begin{array}{l} n, \overline{n} \in \mathbb{M}(N) \text{ and} \\ \kappa = \begin{pmatrix} \text{- -▷- -} \text{ if } \{ n \mapsto \text{- - - -}, \overline{n} \mapsto \text{- - - -} \} \in \mathbb{M}(c) \\ \text{- -◁- -} \text{ if } \{ n \mapsto \text{- - - -}, \overline{n} \mapsto \text{———} \} \in \mathbb{M}(c) \\ \text{———} \text{ if } \{ n \mapsto \text{———}, \overline{n} \mapsto \text{- - - -} \} \in \mathbb{M}(c) \end{pmatrix} \end{array} \right\}$$

$= $ /* By Definition 26 (page 21) of $\mathbb{M}$ for 3-colorings */

$$\left\{ n \mapsto \kappa \;\middle|\; n \in N \text{ and } \kappa = \begin{pmatrix} \text{- -▷- -} \text{ if } c(n) = \text{- -▷- -} \\ \text{- -◁- -} \text{ if } c(n) = \text{- -◁- -} \\ \text{———} \text{ if } c(n) = \text{———} \end{pmatrix} \right\}$$

$= $ /* Because $c$ is a 3-coloring by the premise */

$\{ n \mapsto \kappa \mid n \in N \text{ and } \kappa = c(n) \}$

$= $ /* By Definition 6 (page 7) of colorings */

$c$

**Definition 38 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colorings).** *Let $c$ be an $\mathbb{M}$-ed 3-coloring over $N$. Its $\frac{1}{\mathbb{M}}$-transformation, denoted $\frac{1}{\mathbb{M}}(c)$, is defined as:*

$$\frac{1}{\mathbb{M}}(c) = \left\{ n \mapsto \kappa \;\middle|\; \begin{array}{l} n, \overline{n} \in N \text{ and} \\ \kappa = \begin{pmatrix} \text{- -▷- -} \text{ if } c(n) = \text{- - - -} \text{ and } c(\overline{n}) = \text{- - - -} \\ \text{- -◁- -} \text{ if } c(n) = \text{- - - -} \text{ and } c(\overline{n}) = \text{———} \\ \text{———} \text{ if } c(n) = \text{———} \text{ and } c(\overline{n}) = \text{———} \end{pmatrix} \end{array} \right\}$$

The following lemma states that the application of $\frac{1}{\mathbb{M}}$ to an $\mathbb{M}$-ed 3-coloring yields the untransformed 3-coloring.

**Lemma 18 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colorings inverts $\mathbb{M}$).** *Let $c$ be a 3-coloring over $N$. Then:*

$$\frac{1}{\mathbb{M}}(\mathbb{M}(c)) = c$$

*Proof. Follows from Table 15 (page 41).*

We proceed with the definition of $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-coloring tables. Recall that we defined a coloring table as a set of colorings. Informally, the application of $\frac{1}{\mathbb{M}}$ to an $\mathbb{M}$-ed 3-coloring table yields a set of $\frac{1}{\mathbb{M}}$-ed $\mathbb{M}$-ed 3-colorings.

**Definition 39 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-coloring tables).** *Let $T$ be an $\mathbb{M}$-ed 3-coloring table over $N$. Its $\frac{1}{\mathbb{M}}$-transformation, denoted $\frac{1}{\mathbb{M}}(T)$, is defined as:*

$$\frac{1}{\mathbb{M}}(T) = \{ \tfrac{1}{\mathbb{M}}(c) \mid c \in T \}$$

The following lemma states that the application of $\frac{1}{\mathbb{M}}$ to an $\mathbb{M}$-ed 3-coloring table yields the untransformed 3-coloring table. In our proof, we apply the inversion lemma that concerns $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colorings.

**Table 16.** Proof: $\frac{1}{\mathbb{M}}(\mathbb{M}(T)) = T$

| |
|---|
| $\frac{1}{\mathbb{M}}(\mathbb{M}(T))$ |
| $=$ /* By Definition 27 (page 22) of $\mathbb{M}$ for 3-coloring tables */ |
| $\frac{1}{\mathbb{M}}(\{\, \mathbb{M}(c) \mid c \in T)$ |
| $=$ /* By Definition 39 (page 41) of $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-coloring tables */ |
| $\{\, \frac{1}{\mathbb{M}}(\mathbb{M}(c)) \mid \mathbb{M}(c) \in \{\, \mathbb{M}(c) \mid c \in T \,\} \,\}$ |
| $=$ /* By the inversion of $\mathbb{M}$ for 3-colorings by $\frac{1}{\mathbb{M}}$ in Lemma 18 (page 41) and by Definition 27 (page 22) of $\mathbb{M}$ for 3-coloring tables */ |
| $\{\, c \mid \mathbb{M}(c) \in \mathbb{M}(T) \,\}$ |
| $=$ /* Because, by Definition 27 (page 22) of $\mathbb{M}$ for 3-coloring tables, $\mathbb{M}(c) \in \mathbb{M}(T)$ iff $c \in T$ */ |
| $T$ |

**Lemma 19 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-coloring tables inverts $\mathbb{M}$).** *Let $T$ be a 3-coloring table over $N$. Then:*

$$\frac{1}{\mathbb{M}}(\mathbb{M}(T)) = T$$

*Proof. Follows from Table 16 (page 42)* □

We continue with the definition of $\frac{1}{\mathbb{M}}$ for 3-CTMs. Recall from Definition 9 (page 8) that a CTM maps indexes (representing the states of a connector) to coloring tables (describing the admissible behavior in these states). Informally, the application of $\frac{1}{\mathbb{M}}$ to an $\mathbb{M}$-ed 3-CTM yields a CTM that maps indexes to $\frac{1}{\mathbb{M}}$-ed $\mathbb{M}$-ed 3-coloring tables.

**Definition 40 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-CTMs).** *Let $S$ be an $\mathbb{M}$-ed 3-CTM over $[N, \Lambda]$. Its $\frac{1}{\mathbb{M}}$-transformation, denoted $\frac{1}{\mathbb{M}}(S)$, is defined as:*

$$\frac{1}{\mathbb{M}}(S) = \{\, \lambda \mapsto \frac{1}{\mathbb{M}}(S(\lambda)) \mid \lambda \in \Lambda \,\}$$

The following lemma states that the application of $\frac{1}{\mathbb{M}}$ to an $\mathbb{M}$-ed 3-CTM yields the untransformed 3-CTM. In our proof, we apply the inversion lemma that concerns $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-coloring tables.

**Lemma 20 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-CTMs inverts $\mathbb{M}$).** *Let $S$ be a 3-CTM over $[N, \Lambda]$. Then:*

$$\frac{1}{\mathbb{M}}(\mathbb{M}(S)) = S$$

*Proof. Follows from Table 17 (page 43).* □

We proceed with the definition of $\frac{1}{\mathbb{M}}$ for 3-colored next functions. Recall that a next function maps [index, coloring]-pairs to indexes. Informally, the application of $\frac{1}{\mathbb{M}}$ to an $\mathbb{M}$-ed 3-colored next function yields a next function that maps [index, $\frac{1}{\mathbb{M}}$-ed $\mathbb{M}$-ed 3-coloring]-pairs to indexes.

**Table 17.** Proof: $\frac{1}{\mathbb{M}}(\mathbb{M}(S)) = S$

$\frac{1}{\mathbb{M}}(\mathbb{M}(S))$
$=$ /* By Definition 28 (page 23) of $\mathbb{M}$ for 3-CTMs */
$\frac{1}{\mathbb{M}}(\{\,\lambda \mapsto \mathbb{M}(S(\lambda)) \mid \lambda \in \Lambda\,\})$
$=$ /* By Definition 40 (page 42) of $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-CTMs */
$\{\,\lambda \mapsto \frac{1}{\mathbb{M}}(\mathbb{M}(S(\lambda))) \mid \lambda \in \Lambda\,\}$
$=$ /* By the inversion of $\mathbb{M}$ for 3-coloring tables by $\frac{1}{\mathbb{M}}$ in Lemma 19 (page 42) */
$\{\,\lambda \mapsto S(\lambda) \mid \lambda \in \Lambda\,\}$
$=$ /* Because $S$ is a total map with $\Lambda$ as its domain */
$S$

**Definition 41 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colored next functions).** *Let $\eta$ be an $\mathbb{M}$-ed 3-colored next function over $[N, \Lambda, S]$. Its $\frac{1}{\mathbb{M}}$-transformation, denoted $\frac{1}{\mathbb{M}}(\eta)$, is defined as:*

$$\tfrac{1}{\mathbb{M}}(\eta) = \{\,(\lambda, \tfrac{1}{\mathbb{M}}(c)) \mapsto \eta(\lambda, c)) \mid \lambda \in \Lambda \text{ and } c \in S(\lambda)\,\}$$

The following lemma states that the application of $\frac{1}{\mathbb{M}}$ to an $\mathbb{M}$-ed 3-colored next function yields the untransformed 3-colored next function. In our proof, we apply the inversion lemma that concerns $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colorings.

**Lemma 21 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colored next functions inverts $\mathbb{M}$).** *Let $\eta$ be a 3-colored next function over $[N, \Lambda, S]$. Then:*

$$\tfrac{1}{\mathbb{M}}(\mathbb{M}(\eta)) = \eta$$

*Proof. Follows from Table 18 (page 44)* □

We conclude with the definition of $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colored $\eta$-connectors. Recall from Definition 11 (page 9) that $\eta$-connectors consist of a connector structure and a next function. This allows us to define $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colored $\eta$-connectors in terms of $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed connector structures and $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colored next functions.

**Definition 42 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colored $\eta$-connectors).** *Let $\mathcal{C}^{\mathsf{Col}} = \langle C, \eta\rangle$ be an $\mathbb{M}$-ed 3-colored $\eta$-connector over $[N, \Lambda, S]$. Its $\frac{1}{\mathbb{M}}$-transformation, denoted $\frac{1}{\mathbb{M}}(\mathcal{C}^{\mathsf{Col}})$, is defined as:*

$$\tfrac{1}{\mathbb{M}}(\mathcal{C}^{\mathsf{Col}}) = \langle \tfrac{1}{\mathbb{M}}(C), \tfrac{1}{\mathbb{M}}(\eta)\rangle$$

The following lemma states that the application of $\frac{1}{\mathbb{M}}$ to an $\mathbb{M}$-ed 3-colored $\eta$-connector yields the untransformed 3-colored $\eta$-connector. In our proof, we apply the inversion lemmas that concern $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed connector structures and $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colored next functions.

**Lemma 22 ($\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colored $\eta$-connectors inverts $\mathbb{M}$).** *Let $\mathcal{C}^{\mathsf{Col}} = \langle C, \eta\rangle$ be a 3-colored $\eta$-connector over $[N, \Lambda, S]$. Then:*

**Table 18.** Proof: $\frac{1}{\mathbb{M}}(\mathbb{M}(\eta)) = \eta$

| |
|---|
| $\frac{1}{\mathbb{M}}(\mathbb{M}(\eta))$ |
| $=$ /* By Definition 41 (page 43) of $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colored next functions */ |
| $\{\, \lambda, \frac{1}{\mathbb{M}}(\mathbb{M}(c)) \mapsto (\mathbb{M}(\eta))(\lambda, \mathbb{M}(c)) \mid \lambda \in \Lambda \text{ and } \mathbb{M}(c) \in (\mathbb{M}(S))(\lambda) \,\}$ |
| $=$ /* By Definition 29 (page 24) of $\mathbb{M}$ for 3-colored next functions */ |
| $\{\, \lambda, \frac{1}{\mathbb{M}}(\mathbb{M}(c)) \mapsto \eta(\lambda, c) \mid \lambda \in \Lambda \text{ and } \mathbb{M}(c) \in (\mathbb{M}(S))(\lambda) \,\}$ |
| $=$ /* Because, by Definition 27 (page 22) of $\mathbb{M}$ for 3-coloring tables, $[\mathbb{M}(c) \in (\mathbb{M}(S))(\lambda)]$ iff $[c \in S(\lambda)]$ */ |
| $\{\, \lambda, \frac{1}{\mathbb{M}}(\mathbb{M}(c)) \mapsto \eta(\lambda, c) \mid \lambda \in \Lambda \text{ and } c \in S(\lambda) \,\}$ |
| $=$ /* By the inversion of $\mathbb{M}$ for 3-colorings by $\frac{1}{\mathbb{M}}$ in Lemma 18 (page 41) */ |
| $\{\, \lambda, c \mapsto \eta(\lambda, c) \mid \lambda \in \Lambda \text{ and } c \in S(\lambda) \,\}$ |
| $=$ /* By Definition 10 (page 8) of next functions */ |
| $\eta$ |

$$\tfrac{1}{\mathbb{M}}(\mathbb{M}(\mathcal{C}^{\mathsf{Col}})) = \mathcal{C}^{\mathsf{Col}}$$

*Proof. Applying Definition 30 (page 25) of $\mathbb{M}$ for 3-colored $\eta$-connectors and Definition 42 (page 43) of $\frac{1}{\mathbb{M}}$ for $\mathbb{M}$-ed 3-colored $\eta$-connectors yields:*

$$\left\langle \begin{matrix} \frac{1}{\mathbb{M}}(\mathbb{M}(C)), \\ \frac{1}{\mathbb{M}}(\mathbb{M}(\eta)) \end{matrix} \right\rangle = \left\langle \begin{matrix} C, \\ \eta \end{matrix} \right\rangle \begin{matrix} \text{(I)} \\ \text{(II)} \end{matrix}$$

*Sub-equation* (I) *follows from the inversion of $\mathbb{M}$ for connector structures by $\frac{1}{\mathbb{M}}$ in Lemma 17 (page 40), while sub-equation* (II) *follows from the inversion of $\mathbb{M}$ for 3-colored next functions by $\frac{1}{\mathbb{M}}$ in Lemma 21 (page 43). Therefore, $\frac{1}{\mathbb{M}}(\mathbb{M}(\mathcal{C}^{\mathsf{Col}})) = \mathcal{C}^{\mathsf{Col}}$.* □

## 4 From 2-colored $\eta$-connectors to $\alpha$-connectors

In the previous section, we showed that two coloring models can describe the behavior of context-sensitive connectors. In this section, we proceed this line of work by showing that constraint automata—and in particular the class of port automata—can describe the behavior of context-dependent connectors as well. Our approach, called the $\mathbb{L}$-*transformation*, comprises two unary operators, both denoted by $\mathbb{L}$ for notational convenience, that facilitate the transformation of 2-colored $\eta$-connectors to corresponding $\alpha$-connectors. The $\mathbb{L}$-transformation enables the encoding of context-dependency in PA as follows: starting with a 3-colored $\eta$-connector $\mathcal{C}^{\mathsf{Col}}$, we first generate its corresponding 2-colored $\eta$-connector using $\mathbb{M}$—i.e., $\mathbb{M}(\mathcal{C}^{\mathsf{Col}})$—and then construct the $\alpha$-connector that corresponds to $\mathbb{M}(\mathcal{C}^{\mathsf{Col}})$ using $\mathbb{L}$—i.e., $\mathbb{L}(\mathbb{M}(\mathcal{C}^{\mathsf{Col}}))$. Because $\mathbb{M}$ ensures that the intermediate 2-colored $\eta$-connector $\mathbb{M}(\mathcal{C}^{\mathsf{Col}})$ faithfully describes the possibly context-dependent behavior of the connector that $\mathcal{C}^{\mathsf{Col}}$ models (as proven in the previous section), the $\alpha$-connector $\mathbb{L}(\mathbb{M}(\mathcal{C}^{\mathsf{Col}}))$ does so as well.

We note that the correspondence between 2-colored $\eta$-connectors and $\alpha$-connectors has been argued for informally in several publications [CCA07,Cos10]. We quote from [CCA07]:

> One aspect of the constraint automata model is that transitions in automata are labeled with the collection of nodes that synchronously succeed in a given step, at the exclusion of all other nodes present in the connector being modeled. Calculating this set based on the configuration of a connector (which is equivalent to the state of the constraint automaton) is precisely what connector coloring achieves. That is, the 2-coloring model of a connector produces a set of colorings which can be equated with the transitions in the corresponding constraint automata.

In this section, we present a formal argument.[13] First, we give the definitions of $\mathbb{L}$ in Section 4.1 (page 45). Second, we prove its correctness in Section 4.2 (page 47). Third, we show that $\mathbb{L}$ distributes over composition in Section 4.3 (page 48). Finally, in Section 4.4 (page 52), we discuss an application of $\mathbb{M}$ and $\mathbb{L}$: the verification of context-dependent connectors with Vereofy.

## 4.1 Definitions of $\mathbb{L}$

In this subsection, we define two versions of $\mathbb{L}$—the one operates on 2-colored next functions and the other on 2-colored $\eta$-connectors—that facilitate the transformation of the 2-coloring model of a connector to its corresponding port automaton. Recall from Definition 18 (page 13) that a PA defines a constraint automaton whose all transitions carry $\top$ as data constraint; recall from the same definition that a CA serves as an operational model of the behavior of a connector.

We start with $\mathbb{L}$ for 2-colored next functions. Informally, this unary operator transforms a 2-colored next function over a set of nodes $N$, a set of indexes $\Lambda$, and a 2-coloring table map $S$ to a PA whose set of states equals $\Lambda$. Furthermore, $\mathbb{L}$ includes for each index $\lambda \in \Lambda$ and for each coloring $c \in S(\lambda)$ (recall that CTMs map indexes to coloring tables) a transition labeled with the set of nodes to which $c$ assigns the flow color as firing set; all transitions additionally carry $\top$ as the data constraint.

**Definition 43 ($\mathbb{L}$ for 2-colored next functions).** *Let $\eta$ be a 2-colored next function over $[N, \Lambda, S]$ and $\lambda_0 \in \Lambda$ the index that represents the initial state of the connector whose state-transitions $\eta$ models. Its $\mathbb{L}$-transformation, denoted $\mathbb{L}(\eta)$, is defined as:*

$$\mathbb{L}(\eta) = \langle \Lambda, R, \lambda_0 \rangle$$

*in which:*

$$R = \left\{ \langle \lambda, F, \top, \eta(\lambda, c) \rangle \;\middle|\; \begin{array}{l} \lambda \in \Lambda \text{ and } c \in S(\lambda) \text{ and} \\ F = \{\, n \in N \mid c(n) = \text{——} \,\} \end{array} \right\}$$

[13] Actually, we present a somewhat weaker result: we show only that for each 2-colored $\eta$-connector, there exists a corresponding $\alpha$-connector. We do not require the opposite direction for our current purpose and leave it for future work.

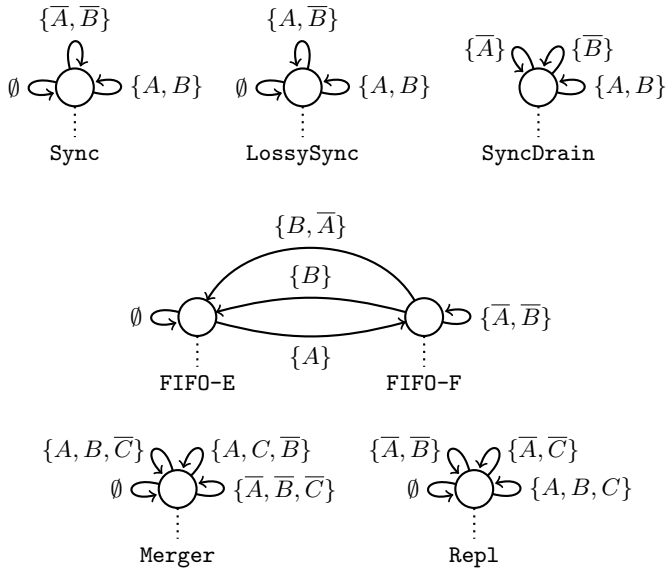


**Fig. 17.** $\mathbb{L}$-ed $\mathbb{M}$-ed 3-coloring models of some of the common primitive connectors.

The following proposition states that the application of $\mathbb{L}$ to a 2-colored next function yields a well-formed PA.

**Proposition 10 ($\mathbb{L}$-ed 2-colored next functions are PA).** *Let $\eta$ be a 2-colored next function over $[N, \Lambda, S]$. Then, $\mathbb{L}(\eta)$ is a PA over $[N]$.*

*Proof.* *By Definition 19 (page 14), we must show that $\mathbb{L}(\eta) = \langle \Lambda, R, \lambda_0 \rangle$ is a CA over $[N, \{\top\}]$. To demonstrate this, by Definition 18 (page 13), we must show that $R \subseteq \Lambda \times 2^N \times \{\top\} \times \Lambda$. Because, by the premise, $\eta$ is a next function over $[N, \Lambda, S]$, by Definition 10 (page 8), the co-domain of $\eta$ is $\Lambda$. Also, by Definition 43 (page 45), for all $\langle \lambda, F, g, \eta(\lambda, c) \rangle \in R$, we have $\lambda \in \Lambda$, $F \subseteq N$, and $g = \top$. Therefore, $\mathbb{L}(\eta) = \langle \Lambda, R, \lambda_0 \rangle$ is a CA over $[N, \{\top\}]$* □

To illustrate the application of $\mathbb{L}$ to 2-colored next functions, in Figure 17 (page 46) and Figure 18 (page 47), we show the $\mathbb{L}$-ed $\mathbb{M}$-ed 3-colored next functions—i.e., $\mathbb{L}$-ed 2-colored next functions, i.e., PA—of some of the common primitive connectors, ExclRouter, and LossyFIFO.[14] Note that the transition $\langle$LosFIFO-E, $\{A\}$, $\top$, LosFIFO-E$\rangle$, which describes the inadmissible behavior in which LossyFIFO loses a data item between $A$ and an empty buffer, does not exist in the context-sensitive PA of LossyFIFO. This contrasts its ordinary PA in Figure 9 (page 15).

We end this subsection with the definition of $\mathbb{L}$ for 2-colored $\eta$-connectors. Recall that $\eta$-connectors consist of a connector structure and a next function. This

[14] We do not depict the transitions that one can induce with the flip-rule. See also Footnote 6 (page 9).

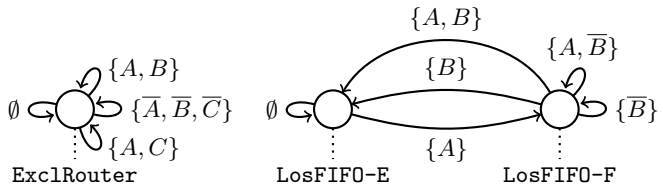


**Fig. 18.** $\mathbb{L}$-ed $\mathbb{M}$-ed 3-coloring models of ExclRouter and LossyFIFO.

allows us to define $\mathbb{L}$ for 2-colored $\eta$-connectors in terms of $\mathbb{L}$ for 2-colored next functions. Note that connector structures do not incur any change; $\mathbb{L}$ changes only the semantic models that we associate these connector structures with.

**Definition 44 ($\mathbb{L}$ for 2-colored $\eta$-connectors).** *Let $\mathcal{C}^{\mathsf{Col}} = \langle C, \eta \rangle$ be a 2-colored $\eta$-connector over $[N, \Lambda, S]$. Its $\mathbb{L}$-transformation, denoted $\mathbb{L}(\mathcal{C}^{\mathsf{Col}})$, is defined as:*

$$\mathbb{L}(\mathcal{C}^{\mathsf{Col}}) = \langle C, \mathbb{L}(\eta) \rangle$$

The following proposition states that the application of $\mathbb{L}$ to a 2-colored $\eta$-connector yields a well-formed $\alpha$-connector.

**Proposition 11 ($\mathbb{L}$-ed 2-colored $\eta$-connectors are $\alpha$-connectors).** *Let $\mathcal{C}^{\mathsf{Col}} = \langle C, \eta \rangle$ be a 2-colored $\eta$-connector over $[N, \Lambda, S]$. Then, $\mathbb{L}(\mathcal{C}^{\mathsf{Col}}) = \langle \mathcal{C}, \mathbb{L}(\eta) \rangle$ is an $\alpha$-connector over $N$.*

*Proof. By Definition 19 (page 14) of $\alpha$-connectors, we must show that $\mathbb{L}(\eta)$ is a PA over $N$. This follows from Proposition 10 (page 46). Therefore, $\mathbb{L}(\mathcal{C}^{\mathsf{Col}})$ is an $\alpha$-connector over $N$.* □

## 4.2 Correctness of $\mathbb{L}$

In this subsection, we show the *correctness* of $\mathbb{L}$: we call the $\mathbb{L}$-transformation correct if, for each 2-colored $\eta$-connector $\mathcal{C}^{\mathsf{Col}}$, its $\mathbb{L}$-transformation $\mathbb{L}(\mathcal{C}^{\mathsf{Col}})$ *bi-simulates* $\mathcal{C}^{\mathsf{Col}}$; the definition of bi-simulation in this context occurs below. As our definition of simulation for $\eta$-connectors, this definition roots in Milner's notion of bi-simulation [Mil89]. Informally, an $\alpha$-connector $\mathcal{C}^{\mathsf{CA}}$ bi-simulates an $\eta$-connector $\mathcal{C}^{\mathsf{Col}}$ (and vice versa: bi-simulation, in contrast to simulation, relates symmetrically) if for each mapping in the next function of $\mathcal{C}^{\mathsf{Col}}$, there exists a corresponding transition in the PA of $\mathcal{C}^{\mathsf{CA}}$, and for each transition in the PA of $\mathcal{C}^{\mathsf{CA}}$, there exists a corresponding mapping in the next function of $\mathcal{C}^{\mathsf{Col}}$.

**Definition 45 (Bi-simulation for $\alpha$-connectors and $\eta$-connectors).** *Let $\mathcal{C}^{\mathsf{CA}} = \langle C, \alpha \rangle$ with $\alpha = \langle Q, R, q_0 \rangle$ be an $\alpha$-connector over $[N, G]$ and $\mathcal{C}^{\mathsf{Col}} = \langle C, \eta \rangle$ an $\eta$-connector over $[N, \Lambda, S]$. $\mathcal{C}^{\mathsf{CA}}$ and $\mathcal{C}^{\mathsf{Col}}$ are bi-similar, denoted $\mathcal{C}^{\mathsf{CA}} \sim \mathcal{C}^{\mathsf{Col}}$, if there exists a relation $\mathcal{R} \subseteq Q \times \Lambda$ such that, for all $\langle q, \lambda \rangle \in \mathcal{R}$:*

(I) *If* $\langle q, F, \top, q' \rangle \in R$ *then there exists a* $\lambda' \in \Lambda$ *such that:*

- $[\langle \lambda, c \rangle \mapsto \lambda'] \in \eta$;
- $\langle \lambda', q' \rangle \in \mathcal{R}$;
- $F = \{\, n \in N \mid c(n) = \text{——} \,\}$.

(II) *If* $[\langle \lambda, c \rangle \mapsto \lambda'] \in \eta$ *then there exists a* $q' \in Q$ *such that:*

- $\langle q, F, \top, q' \rangle \in R$;
- $\langle \lambda', q' \rangle \in \mathcal{R}$;
- $F = \{\, n \in N \mid c(n) = \text{——} \,\}$.

*In that case,* $\mathcal{R}$ *is called a bi-simulation relation.*

The following lemma establishes the correctness of $\mathbb{L}$: it states that the $\mathbb{L}$-transformation of a 2-colored $\eta$-connector bi-simulates this 2-colored $\eta$-connector (and, by symmetry, vice versa).

**Lemma 23 ($\mathbb{L}$-ed 2-colored $\eta$-connectors and 2-colored $\eta$-connectors are bi-similar).** *Let* $\mathcal{C}^{\mathsf{Col}} = \langle \mathcal{C}, \eta \rangle$ *be a 2-colored* $\eta$*-connector over* $[N, \Lambda, S]$. *Then,* $\mathbb{L}(\mathcal{C}^{\mathsf{Col}}) \sim \mathcal{C}^{\mathsf{Col}}$.

*Proof. Let* $\mathcal{R} = \{\, \langle \lambda, \lambda \rangle \mid \lambda \in \Lambda \,\}$. *We show that* $\mathcal{R}$ *is a bi-simulation relation by demonstrating that it satisfies* (I) *and* (II) *of Definition 45 (page 47). Let* $\langle \lambda, \lambda \rangle \in \mathcal{R}$.

(I) *Suppose* $\langle \lambda, F, \top, \lambda' \rangle \in R$. *Then, by Definition 43 (page 45) of* $R$, *there exists a* $c \in S(\lambda)$ *such that* $\lambda' = \eta(\lambda, c)$ *and* $\lambda' \in \Lambda$. *Hence,* $[\langle \lambda, c \rangle \mapsto \lambda'] \in \eta$. *Also, by the definition of* $\mathcal{R}$, $\langle \lambda', \lambda' \rangle \in \mathcal{R}$. *Finally, by Definition 43 (page 45) of* $R$, $F = \{\, n \in N \mid c(n) = \text{——} \,\}$. *Therefore,* $\mathcal{R}$ *satisfies* (I).

(II) *Suppose* $[\langle \lambda, c \rangle \mapsto \lambda'] \in \eta$. *Then, by Definition 43 (page 45) of* $R$, *there exists a* $c \in S(\lambda)$ *such that* $\langle \lambda, F, \top, \lambda' \rangle \in R$ *and* $\lambda' \in \Lambda$. *Also, by the definition of* $\mathcal{R}$, $\langle \lambda', \lambda' \rangle \in \mathcal{R}$. *Finally, by Definition 43 (page 45) of* $R$, $F = \{\, n \in N \mid c(n) = \text{——} \,\}$. *Therefore,* $\mathcal{R}$ *satisfies* (II).

*Thus,* $\mathcal{R}$ *satisfies* (I) *and* (II). *Hence,* $\mathcal{R}$ *is a bi-simulation relation. Therefore,* $\mathbb{L}(\mathcal{C}^{\mathsf{Col}}) \sim \mathcal{C}^{\mathsf{Col}}$. □

### 4.3 Distributivity of $\mathbb{L}$

In this subsection, we demonstrate that $\mathbb{L}$ distributes over composition, important for similar reasons as the relevance of the distributivity of $\mathbb{M}$ over composition in Section 3.3 (page 30): we prefer (i) to compute $\mathbb{L}$-transformations only once for the common primitive connectors and (ii) to compose the resulting port automata to form more complex PA. Because we defined $\mathbb{L}$ for two types of operands (2-colored next functions and 2-colored $\eta$-connectors), this subsection consists of two lemmas: one distributivity lemma for each form of $\mathbb{L}$.

We start with the distributivity lemma of $\mathbb{L}$ for 2-colored next functions. Informally, this lemma states that it does not matter whether we first compose 2-colored next functions $\eta_1$ and $\eta_2$ and then apply $\mathbb{L}$ to the composition or first apply $\mathbb{L}$ to $\eta_1$ and $\eta_2$ and then compose the transformations; the resulting PA equal each other. Our proof, although rather technical and relatively long, essentially comprises of a series of straightforward applications of definitions that allow us to rewrite the transition relations of the composed PA.

**Lemma 24 ($\mathbb{L}$ for 2-colored next functions distributes over composition).** *Let $\eta_1$ and $\eta_2$ be 2-colored next functions over $[N_1, \Lambda_1, S_1]$ and $[N_2, \Lambda_2, S_2]$. Additionally, let $\lambda_0^1$ and $\lambda_0^2$ be indexes representing the initial states of the connectors whose state-transitions $\eta_1$ and $\eta_2$ model. Then:*

$$\mathbb{L}(\eta_1) \bowtie \mathbb{L}(\eta_2) = \mathbb{L}(\eta_1 \otimes \eta_2)$$

*Proof. Applying Definition 43 (page 45) of $\mathbb{L}$ for 2-colored next functions to the left-hand side (LHS) and Definition 15 (page 12) of composition for next functions to the right-hand side (RHS) yields:*

$$\begin{array}{c}\langle \Lambda_1, R_1, \lambda_0^1 \rangle \\ \bowtie \\ \langle \Lambda_2, R_2, \lambda_0^2 \rangle\end{array} = \mathbb{L}\left(\left\{ \begin{array}{c} \langle \lambda_1, \lambda_2 \rangle, c_1 \cup c_2 \\ \downarrow \\ \langle \eta_1(\lambda_1, c_1), \eta_2(\lambda_2, c_2) \rangle \end{array} \middle| \begin{array}{l} \langle \lambda_1, \lambda_2 \rangle \in \Lambda_1 \times \Lambda_2 \text{ and} \\ c_1 \cup c_2 \in (S_1 \odot S_2)(\langle \lambda_1, \lambda_2 \rangle) \end{array} \right\}\right)$$

*in which:*

$$R_1 = \left\{ \langle \lambda_1, F_1, \top, \eta_1(\lambda_1, c_1) \rangle \middle| \begin{array}{l} \lambda_1 \in \Lambda_1 \text{ and } c_1 \in S_1(\lambda_1) \text{ and} \\ F_1 = \{\, n \in N_1 \,|\, c_1(n) = \text{——} \,\} \end{array} \right\}$$

*and:*

$$R_2 = \left\{ \langle \lambda_2, F_2, \top, \eta_2(\lambda_2, c_2) \rangle \middle| \begin{array}{l} \lambda_2 \in \Lambda_2 \text{ and } c_2 \in S_2(\lambda_2) \text{ and} \\ F_2 = \{\, n \in N_2 \,|\, c_2(n) = \text{——} \,\} \end{array} \right\}$$

*Applying Definition 20 (page 14) of composition for CA to the LHS, and Definition 43 (page 45) of $\mathbb{L}$ for next functions to the RHS yields:*

$$\langle \Lambda_1 \times \Lambda_2, R, \langle \lambda_0^1, \lambda_0^2 \rangle \rangle = \langle \Lambda_1 \times \Lambda_2, R', \langle \lambda_0^1, \lambda_0^2 \rangle \rangle$$

*in which:*

$$R = \left\{ \langle \langle \lambda_1, \lambda_2 \rangle, F_1 \cup F_2, \top, \langle \lambda_1', \lambda_2' \rangle \rangle \middle| \begin{array}{l} \langle \lambda_1, F_1, \top, \lambda_1' \rangle \in R_1 \\ \text{and } \langle \lambda_2, F_2, \top, \lambda_2' \rangle \in R_2 \\ \text{and } F_1 \cap N_2 = F_2 \cap N_1 \end{array} \right\}$$

*and:*

$$R' = \left\{ \langle \lambda, F, \top, (\eta_1 \otimes \eta_2)(\lambda, c) \rangle \middle| \begin{array}{l} \lambda \in \Lambda_1 \times \Lambda_2 \text{ and } c \in (S_1 \odot S_2)(\lambda) \\ \text{and } F = \{\, n \in N_1 \cup N_2 \,|\, c(n) = \text{——} \,\} \end{array} \right\}$$

*What remains to be shown is $R = R'$. This follows from Table 19 (page 50) and Table 20 (page 51). Therefore, $\mathbb{L}(\eta_1) \bowtie \mathbb{L}(\eta_2) = \mathbb{L}(\eta_1 \otimes \eta_2)$.* □

We marked the most interesting step in our proof with "$\doteq$" (instead of the ordinary "="), which occurs as the fourth equality symbol (counted from top to bottom) in Table 19 (page 50). In that step, we use the premise that the *2*-colored next functions under consideration map only *2*-colorings. In case of 3-colored next functions, this step becomes invalid, and the proof breaks down.

We end this subsection with the distributivity lemma of $\mathbb{L}$ for 2-colored $\eta$-connectors. Informally, this lemma states that it does not matter whether we first

**Table 19.** Proof: $R = R'$ (first part)

$$
\begin{array}{ll}
 & R \\
= & /^{*} \text{ By the definition of } R \text{ in Lemma 24 (page 49) } {}_{*}/ \\
 & \left\{ \langle\langle \lambda_1, \lambda_2\rangle, F_1 \cup F_2, \top, \langle \lambda_1', \lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \langle \lambda_1, F_1, \top, \lambda_1'\rangle \in R_1 \text{ and } \langle \lambda_2, F_2, \top, \lambda_2'\rangle \in R_2 \\ \text{and } F_1 \cap N_2 = F_2 \cap N_1 \end{array} \right\} \\
= & /^{*} \text{ By the definitions of } R_1 \text{ and } R_2 \text{ in Lemma 24 (page 49) } {}_{*}/ \\
 & \left\{ \langle\langle \lambda_1, \lambda_2\rangle, F_1 \cup F_2, \top, \langle \lambda_1', \lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \lambda_1 \in \Lambda_1 \text{ and } c_1 \in S_1(\lambda_1) \text{ and } \lambda_1' = \eta_1(\lambda_1, c_1) \\ \text{and } F_1 = \{\, n \in N_1 \,|\, c_1(n) = \text{——} \,\} \text{ and} \\ \lambda_2 \in \Lambda_2 \text{ and } c_2 \in S_2(\lambda_2) \text{ and } \lambda_2' = \eta_2(\lambda_2, c_2) \\ \text{and } F_2 = \{\, n \in N_2 \,|\, c_2(n) = \text{——} \,\} \\ \text{and } F_1 \cap N_2 = F_2 \cap N_1 \end{array} \right\} \\
= & /^{*} \text{ Because, by the definition of the Cartesian product, } [\lambda_1 \in \Lambda_1 \text{ and } \lambda_2 \in \Lambda_2] \text{ iff } \langle \lambda_1, \lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \; {}_{*}/ \\
 & \left\{ \langle\langle \lambda_1, \lambda_2\rangle, F_1 \cup F_2, \top, \langle \lambda_1', \lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \langle \lambda_1, \lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \\ \text{and } c_1 \in S_1(\lambda_1) \text{ and } c_2 \in S_2(\lambda_2) \\ \text{and } \lambda_1' = \eta_1(\lambda_1, c_1) \text{ and } \lambda_2' = \eta_2(\lambda_2, c_2) \\ \text{and } F_1 = \{\, n \in N_1 \,|\, c_1(n) = \text{——} \,\} \\ \text{and } F_2 = \{\, n \in N_2 \,|\, c_2(n) = \text{——} \,\} \\ \text{and } F_1 \cap N_2 = F_2 \cap N_1 \end{array} \right\} \\
\doteq & /^{*} \text{ Because, by the definition of } F_1 \text{ and } F_2 \text{ in Lemma 24 (page 49), } [F_1 \cap N_2 = F_2 \cap N_1] \text{ iff } [\{\, n \in N_1 \cap N_2 \,|\, c_1(n) = \text{——} \,\} = \{\, n \in N_1 \cap N_2 \,|\, c_2(n) = \text{——} \,\}], \text{ and because, since } c_1 \text{ and } c_2 \text{ are 2-colorings, } [\{\, n \in N_1 \cap N_2 \,|\, c_1(n) = \text{——} \,\} = \{\, n \in N_1 \cap N_2 \,|\, c_2(n) = \text{——} \,\}] \text{ iff } [c_1(n) = c_2(n) \text{ for all } n \in N_1 \cap N_2] \; {}_{*}/ \\
 & \left\{ \langle\langle \lambda_1, \lambda_2\rangle, F_1 \cup F_2, \top, \langle \lambda_1', \lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \langle \lambda_1, \lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \\ \text{and } c_1 \in S_1(\lambda_1) \text{ and } c_2 \in S_2(\lambda_2) \\ \text{and } \lambda_1' = \eta_1(\lambda_1, c_1) \text{ and } \lambda_2' = \eta_2(\lambda_2, c_2) \\ \text{and } F_1 = \{\, n \in N_1 \,|\, c_1(n) = \text{——} \,\} \\ \text{and } F_2 = \{\, n \in N_2 \,|\, c_2(n) = \text{——} \,\} \\ \text{and } c_1(n) = c_2(n) \text{ for all } n \in N_1 \cap N_2 \end{array} \right\} \\
= & /^{*} \text{ Because, by Definition 13 (page 10) of composition for coloring tables, } [c_1 \in S(\lambda_1) \text{ and } c_2 \in S(\lambda_2) \text{ and } c_1(n) = c_2(n) \text{ for all } n \in N_1 \cap N_2] \text{ iff } [c_1 \cup c_2 \in S(\lambda_1) \cdot S(\lambda_2)] \; {}_{*}/ \\
 & \left\{ \langle\langle \lambda_1, \lambda_2\rangle, F_1 \cup F_2, \top, \langle \lambda_1', \lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \langle \lambda_1, \lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \\ \text{and } c_1 \cup c_2 \in S_1(\lambda_1) \cdot S_2(\lambda_2) \\ \text{and } \lambda_1' = \eta_1(\lambda_1, c_1) \text{ and } \lambda_2' = \eta_2(\lambda_2, c_2) \\ \text{and } F_1 = \{\, n \in N_1 \,|\, c_1(n) = \text{——} \,\} \\ \text{and } F_2 = \{\, n \in N_2 \,|\, c_2(n) = \text{——} \,\} \end{array} \right\} \\
= & /^{*} \text{ Continued in Table 20 } {}_{*}/ \\
 & \ldots
\end{array}
$$

**Table 20.** Proof: $R = R'$ (second part)

$$\left\{ \langle\langle \lambda_1, \lambda_2\rangle, F_1 \cup F_2, \top, \langle \lambda_1', \lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \langle \lambda_1, \lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \\ \text{and } c_1 \cup c_2 \in S_1(\lambda_1) \cdot S_2(\lambda_2) \\ \text{and } \lambda_1' = \eta_1(\lambda_1, c_1) \text{ and } \lambda_2' = \eta_2(\lambda_2, c_2) \\ \text{and } F_1 = \{\, n \in N_1 \,|\, c_1(n) = \text{———} \,\} \\ \text{and } F_2 = \{\, n \in N_2 \,|\, c_2(n) = \text{———} \,\} \end{array} \right\}$$

$=$ /* By Definition 14 (page 11) of composition for coloring table maps */

$$\left\{ \langle\langle \lambda_1, \lambda_2\rangle, F_1 \cup F_2, \top, \langle \lambda_1', \lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \langle \lambda_1, \lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \\ \text{and } c_1 \cup c_2 \in (S_1 \odot S_2)(\langle \lambda_1, \lambda_2\rangle) \\ \text{and } \lambda_1' = \eta_1(\lambda_1, c_1) \text{ and } \lambda_2' = \eta_2(\lambda_2, c_2) \\ \text{and } F_1 = \{\, n \in N_1 \,|\, c_1(n) = \text{———} \,\} \\ \text{and } F_2 = \{\, n \in N_2 \,|\, c_2(n) = \text{———} \,\} \end{array} \right\}$$

$=$ /* Because, by Definition 15 (page 12) of composition for next functions, $[\langle \lambda_1, \lambda_2\rangle \in \Lambda_1 \times \Lambda_2$ and $c_1 \cup c_2 \in (S_1 \odot S_2)(\lambda)$ and $\lambda_1' = \eta_1(\lambda_1, c_1)$ and $\lambda_2' = \eta_2(\lambda_2, c_2)]$ iff $[\langle \lambda_1', \lambda_2'\rangle = (\eta_1 \otimes \eta_2)(\langle \lambda_1, \lambda_2\rangle, c_1 \cup c_2)]$ */

$$\left\{ \langle\langle \lambda_1, \lambda_2\rangle, F_1 \cup F_2, \top, \langle \lambda_1', \lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \langle \lambda_1, \lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \\ \text{and } c_1 \cup c_2 \in (S_1 \odot S_2)(\langle \lambda_1, \lambda_2\rangle) \\ \text{and } \langle \lambda_1', \lambda_2'\rangle = (\eta_1 \otimes \eta_2)(\langle \lambda_1, \lambda_2\rangle, c_1 \cup c_2) \\ \text{and } F_1 = \{\, n \in N_1 \,|\, c_1(n) = \text{———} \,\} \\ \text{and } F_2 = \{\, n \in N_2 \,|\, c_2(n) = \text{———} \,\} \end{array} \right\}$$

$=$ /* By applying $F = F_1 \cup F_2$, and by the definitions of $F_1$ and $F_2$ in Lemma 24 (page 49) */

$$\left\{ \langle\langle \lambda_1, \lambda_2\rangle, F, \top, \langle \lambda_1', \lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \langle \lambda_1, \lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \\ \text{and } c_1 \cup c_2 \in (S_1 \odot S_2)(\langle \lambda_1, \lambda_2\rangle) \\ \text{and } \langle \lambda_1', \lambda_2'\rangle = (\eta_1 \otimes \eta_2)(\langle \lambda_1, \lambda_2\rangle, c_1 \cup c_2) \\ \text{and } F = \left\{ n \in N_1 \cup N_2 \;\middle|\; \begin{array}{l} c_1(n) = \text{———} \\ \text{or } c_2(n) = \text{———} \end{array} \right\} \end{array} \right\}$$

$=$ /* Because, by Definition 12 (page 10) of composition for colorings, $[c_1(n) =$ ——— or $c_2(n) =$ ———$]$ iff $[c_1 \cup c_2 =$ ———$]$ */

$$\left\{ \langle\langle \lambda_1, \lambda_2\rangle, F, \top, \langle \lambda_1', \lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \langle \lambda_1, \lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \\ \text{and } c_1 \cup c_2 \in (S_1 \odot S_2)(\langle \lambda_1, \lambda_2\rangle) \\ \text{and } \langle \lambda_1', \lambda_2'\rangle = (\eta_1 \otimes \eta_2)(\langle \lambda_1, \lambda_2\rangle, c_1 \cup c_2) \\ \text{and } F = \{\, n \in N_1 \cup N_2 \,|\, c_1 \cup c_2 = \text{———} \,\} \end{array} \right\}$$

$=$ /* By applying $\lambda = \langle \lambda_1, \lambda_2\rangle$ and $c = c_1 \cup c_2$ */

$$\left\{ \langle \lambda, F, \top, (\eta_1 \otimes \eta_2)(\lambda, c)\rangle \;\middle|\; \begin{array}{l} \lambda \in \Lambda_1 \times \Lambda_2 \text{ and } c \in (S_1 \odot S_2)(\lambda) \\ \text{and } F = \{\, n \in N_1 \cup N_2 \,|\, c_1 \cup c_2 = \text{———} \,\} \end{array} \right\}$$

$=$ /* By definition of $R'$ */

$R'$

compose 2-colored $\eta$-connectors $\mathcal{C}_1^{\mathsf{Col}}$ and $\mathcal{C}_2^{\mathsf{Col}}$ and then apply $\mathbb{L}$ to the composition or first apply $\mathbb{L}$ to $\mathcal{C}_1^{\mathsf{Col}}$ and $\mathcal{C}_2^{\mathsf{Col}}$ and then compose the transformations; the resulting $\alpha$-connectors equal each other. In our proof, we apply the distributivity lemma that concerns $\mathbb{L}$ for 2-colored next functions.

**Lemma 25 ($\mathbb{L}$ for 2-colored $\eta$-connectors distributes over composition).** *Let $\mathcal{C}_1^{\mathsf{Col}} = \langle C_1, \eta_1 \rangle$ and $\mathcal{C}_2^{\mathsf{Col}} = \langle C_2, \eta_2 \rangle$ be 2-colored $\eta$-connectors over $[N_1, \Lambda_1, S_1]$ and $[N_2, \Lambda_2, S_2]$. Additionally, let $\lambda_0^1$ and $\lambda_0^2$ be indexes representing the initial states of the connectors that $\mathcal{C}_1^{\mathsf{Col}}$ and $\mathcal{C}_2^{\mathsf{Col}}$ model. Then:*

$$\mathbb{L}(\mathcal{C}_1^{\mathsf{Col}}) \times \mathbb{L}(\mathcal{C}_2^{\mathsf{Col}}) = \mathbb{L}(\mathcal{C}_1^{\mathsf{Col}} \times \mathcal{C}_2^{\mathsf{Col}})$$

*Proof. Applying Definition 44 (page 47) of $\mathbb{L}$ for 2-colored $\eta$-connectors, Definition 16 (page 12) of composition for $\eta$-connectors, and Definition 21 (page 15) of composition for $\alpha$-connectors yields:*

$$\left\langle \begin{matrix} C_1 \boxtimes C_2, \\ \mathbb{L}(\eta_1) \bowtie \mathbb{L}(\eta_2) \end{matrix} \right\rangle = \left\langle \begin{matrix} C_1 \boxtimes C_2, \\ \mathbb{L}(\eta_1 \otimes \eta_2) \end{matrix} \right\rangle \begin{matrix} \text{(I)} \\ \text{(II)} \end{matrix}$$

*Sub-equation* (I) *follows trivially, while sub-equation* (II) *follows from the distributivity of $\mathbb{L}$ for 2-colored next functions over composition in Lemma 24 (page 49). Therefore, $\mathbb{M}(\mathcal{C}_1^{\mathsf{Col}}) \times \mathbb{M}(\mathcal{C}_2^{\mathsf{Col}}) = \mathbb{M}(\mathcal{C}_1^{\mathsf{Col}} \times \mathcal{C}_2^{\mathsf{Col}})$.* □

## 4.4 Application: model checking with Vereofy

We proceed with an application of $\mathbb{M}$ and $\mathbb{L}$: the verification of context-sensitive connectors with Vereofy [BBKK09b,BBKK09a],[15] a model checking tool for the analysis of Reo connectors based on constraint automata. Vereofy, developed by the group of Christel Baier at the Technical University of Dresden, allows for the verification of temporal properties expressed in LTL and CTL-like logics, and it supports bi-simulation equivalence checks. Moreover, it can generate counterexamples and provides a GUI integration with the Eclipse Coordination Tools.[16]

Up to now, Vereofy could not faithfully model check context-dependent connectors; to verify such connectors, one needed to implement context-dependency as non-determinism, potentially rendering the result of a verification run meaningless. Using $\mathbb{M}$ and $\mathbb{L}$, however, we mend this deficiency: the port automata that result from applying our transformation operators properly describe the behavior of context-sensitive connectors, and since PA constitute a subclass of CA, we can specify and verify them with Vereofy. To summarize our approach:

1. Transform the common primitive 3-colored $\eta$-connectors to context-dependent 2-colored $\eta$-connectors using $\mathbb{M}$. Examples appear in Figure 13 (page 23), Figure 14 (page 23), and Figure 16 (page 26).

[15] Vereofy homepage: http://www.vereofy.de
[16] Eclipse Coordination Tools homepage: http://reo.project.cwi.nl

2. Transform these context-dependent 2-colored $\eta$-connectors to context-dependent $\alpha$-connectors using $\mathbb{L}$. Examples appear in Figure 17 (page 46).
3. Compose these context-dependent $\alpha$-connectors to construct more complex connectors. Examples appear in Figure 18 (page 47).

Although simple and straightforward, this recipe enables the analysis of context-sensitive connectors with Vereofy in *many* cases. Unfortunately, there exist cases in which this approach does not work due to the presence of causality loops: since there do not exist methods for the detection and removal of causality loops from CA, we must remove these from the coloring models of connectors—i.e., *before* we transform coloring models to CA. To summarize our approach in this case:

1. Transform the common primitive 3-colored $\eta$-connectors to *constructive 3-colored $\eta$-connectors* that consist of *constructive 3-colorings* [Cos10] to facilitate the detection and removal of causality loops.
2. Compose these constructive 3-colored $\eta$-connectors to construct more complex constructive 3-colored $\eta$-connectors.
3. Detect and remove causality loops.
4. Transform these filtered constructive 3-colored $\eta$-connectors to 2-colored $\eta$-connectors using $\mathbb{M}$.
5. Transform these 2-colored $\eta$-connectors to $\alpha$-connectors using $\mathbb{L}$.

Shortly, we illustrate both of the two approaches with an example.

Vereofy comes with a library that contains CA of most of the common primitive connectors, written in two input languages: the *Reo Scripting Language* (a textual version of Reo) and the guarded command language *CARML* (a textual version of constraint automata). For our purpose, we adapted and extended this library: using $\mathbb{M}$ and $\mathbb{L}$, we wrote a new library `CD-Library` that contains context-dependent versions of some of the common primitive connectors.[17] Specifically, `CD-Library` contains the context-dependent CA of Sync, LossySync, FIFO, SyncDrain, Merger, and Replicator. In addition to these primitive connectors, we also included atomic context-dependent CA of ExclRouter and SyncFIFO (the composed versions of these CA contain causality loops).

In the rest of this subsection, we give two examples of the analysis of context-dependent connectors with Vereofy: we start with the specification and verification of LossyFIFO and end with the specification and verification of SyncFIFO.

**LossyFIFO** A simple example concerns LossyFIFO. In Figure 19 (page 54), we repeat its ordinary port automaton (on the left) and the PA that results from applying $\mathbb{M}$ and $\mathbb{L}$ to its 3-coloring model (on the right). In the latter, for simplicity, we hide the context nodes $\overline{A}$ and $\overline{B}$. (Note, however, that for the proper composition of LossyFIFO with other connectors, one requires an instance with visible context nodes.) Additionally, in Figure 19 (page 54), we show the RSL code that specifies these automata.

[17] `CD-Library` and examples: http://reo.project.cwi.nl/vereofy_CD.tar.gz

```
1  CIRCUIT LOSSY_FIFO_ND {
2
3    // Channels
4    new LOSSY_SYNC_ND(A;M);
5    new FIFO1(M;B);
6
7    // Boundary
8    source[0]=A;
9    sink[0]=B;
10
11   // Hide
12   M=NULL;
13 }
```

```
1  CIRCUIT LOSSY_FIFO_CD {
2
3    // Channels
4    new LOSSY_SYNC_CD(A,nM;M,nA);
5    new FIFO1_CD(M,nB;B,nM);
6
7    // Boundary
8    source[0]=A; source[1]=nB;
9    sink[0]=B; sink[1]=nA;
10
11   // Hide
12   M=NULL; nM=NULL;
13 }
```

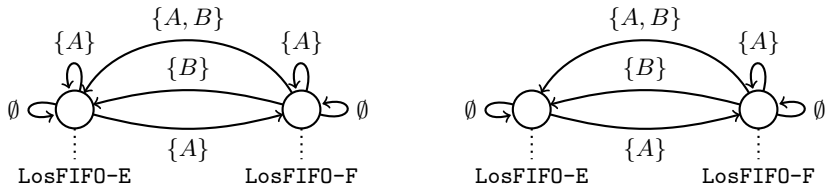


**Fig. 19.** Ordinary PA of LossySync (bottom–left) and its specification in RSL (top–left) and the $\mathbb{L}$-ed $\mathbb{M}$-ed 3-coloring model of LossySync (bottom–right) and its specification in RSL (top–right).

The two CA equal each other except for one transition of the ordinary CA: $\langle$LosFIFO-E, $\{A\}, \top$, LosFIFO-E$\rangle$. This transition describes an inadmissible behavior of LossyFIFO, namely the loss of a data item between $A$ and an empty buffer. In contrast, the CA that results from applying $\mathbb{M}$ and $\mathbb{L}$ to the 3-coloring model of LossyFIFO does not have this transition and thus, does not permit this inadmissible behavior. The following LTL property demonstrates this difference: $\Box((\texttt{LosFIFO-E} \wedge A) \rightarrow \bigcirc \texttt{LosFIFO-F})$. This property states that always if LossyFIFO has an empty buffer and its input node $A$ fires, it has a full buffer one step forward in time. Whereas the ordinary CA violates this property, the context-dependent CA satisfies it.

**SyncFIFO** A more complex example concerns SyncFIFO. In fact, a first attempt to model SyncFIFO using our library of context-sensitive primitive connectors failed due to the presence of causality loops in the resulting composition. Because we cannot detect and remove causality loops from constraint automata, we had to handle these in the 3-coloring model of the connector (using constructive colorings) as described above. In Figure 20 (page 55), we depict the ordinary CA of SyncFIFO and the context-sensitive CA that results from this procedure. At first sight, these automata seem very similar. In fact, if we hide all context nodes in the context-dependent CA on the right, we obtain two identical automata.

The crux of the difference between the two automata, therefore, lies exactly in these context nodes: in contrast to LossyFIFO, SyncFIFO itself exhibits context-dependent behavior (instead of only the primitive connectors that constitute it, namely LossySync). Recall that in the EMPTY state, if the output node of

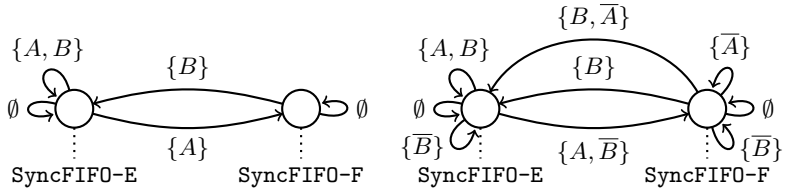


**Fig. 20.** Ordinary PA of SyncFIFO (left) and its $\mathbb{L}$-ed $\mathbb{M}$-ed 3-coloring model (right).

SyncFIFO (henceforth: $B$) lacks a take request, a write request on its input node (henceforth: $A$) causes a data item to flow into the buffer. However, if $B$ has a pending take request, a write request on $A$ causes a data item to flow immediately to node $B$. The ordinary CA of SyncFIFO does not capture this difference, which means that an implementation of this CA would non-deterministically choose one of these two options in case of a pending write request on $A$ and a pending take request on $B$. In contrast, an implementation of the context-dependent CA of SyncFIFO always chooses the appropriate option, because in the absence of a take requests on $B$, data items with irrelevant content—i.e., *signals*—flow through $\overline{B}$. To illustrate this, we encourage the interested reader to compose SyncFIFO and FIFO in the same way we composed LossySync and FIFO.[18]

## 5 Related work

In [AR03], Arbab et al. introduce a coalgebra-based semantic model for Reo. Some years later, in [BSAR06], Baier et al. present an automaton-based approach, namely constraint automata (CA), and prove correspondences with the coalgebra-based model. In [CCA07], however, Clarke et al. observe that neither of these models can handle context-sensitivity, and they introduce the 2-coloring and 3-coloring models to mend this deficiency. Since then, other semantic models with the same aim have come to existence. In [Cos10], Costa introduces *intentional automata* (IA) as an operational model with constructs

[18] The CA that results from composing the ordinary CA of SyncFIFO and FIFO includes a transition that describes an inadmissible behavior in which, in case both SyncFIFO and FIFO have empty buffers, the input node of SyncFIFO fires and causes its own buffer to become full (while the buffer of FIFO remains empty). The CA that results from composing the context-dependent CA of SyncFIFO and FIFO, in contrast, does not include such a transition: if the input node of SyncFIFO fires when both SyncFIFO and FIFO have empty buffers, the buffer of FIFO becomes full (while the buffer of SyncFIFO remains empty).

In Vereofy, we can verify this by model checking both connectors for the property $\square((\texttt{SyncFIFOFIFO-EE} \wedge A) \rightarrow \bigcirc\, \texttt{SyncFIFOFIFO-EF})$, where `SyncFIFOFIFO-EE` represents the state with two empty buffers, and `SyncFIFOFIFO-EF` represents the state with one full buffer (namely FIFO's) and one empty buffer (namely SyncFIFO's). Whereas the ordinary CA violates this property, the context-dependent CA satisfies it.

for context-dependency. Unlike CA, whose states correspond one-to-one to the internal configurations of connectors, IA have more states than the connectors they model; each state of an IA contains information about not only the configuration of the connector, but also about the nodes that *intend* to fire (i.e., with a pending I/O-request). Similarly, transition-labels consist of two sets of nodes: those that intend to fire, and those that actually fire. By maintaining information about I/O-request on nodes, IA capture context-dependency. The number of states, however, quickly grows large, whereas our approach yields succinct CA. In [BCS09], Bonsangue et al. introduce *guarded automata* (GA) as another automaton-based model for capturing context-dependency. Like CA, the states of GA correspond one-to-one to the configurations of connectors, which makes them significantly more compact than IA. To encode context-sensitivity, every transition-label of a GA consists of a *guard* and a *string*. Together, they express which nodes can fire (the string), given the presence and absence of requests at certain nodes (the guard). Guarded automata seem very similar to the CA we obtain with our approach: instead of guards that contain negative occurrences of (base) nodes to specify that these nodes have no pending I/O-requests, we make these negative occurrences explicit with the introduction of (flow through) context nodes.

Besides Vereofy, other approaches to model checking Reo connectors exist. In [KKdV10], Kokash et al. employ the mCRL2 toolset, developed at the TU of Eindhoven, for model checking connectors, combined with a translation tool that automatically generates mCRL2 specifications from graphical models of Reo connectors. The tool's original algorithm operated on constraint automata, making it impossible to verify context-dependent connectors using this approach. Later, however, Kokash et al. incorporated (3-)coloring information in the tool, thus facilitating verification of context-dependent connectors. This advantage of mCRL2 over Vereofy, which could not handle context-dependent connectors up to now, seems no longer valid as we have shown how to encode context-sensitivity in Vereofy. An advantage of Vereofy over mCRL2, on the other hand, is its ability to generate *counterexamples*, which mCRL2 cannot do. In [Kem09], Kemper introduces a SAT-based approach to model checking *timed constraint automata* (TCA). In her work, Kemper represents TCA as formulas in propositional logic and uses existing SAT solvers for verification. This approach allows for model checking timed properties of Reo connectors, but it cannot handle context-dependency. In [MSA06], Mousavi et al. develop a structural operational semantics in Plotkin's style for Reo, encode this semantics in the Maude term-rewriting language, and use Maude's LTL model checking module to verify Reo connectors. In [KSA+08], Khosravi et al. introduce a mapping from Reo to Alloy, a modeling language based on first-order relational logic, and apply the Alloy Analyzer for verification. Although the approach can handle some context-dependent connectors—using a *maximal progress rule* that removes undesired behavior—Khosravi et al. admit to have considerable performance issues.

## 6 Concluding remarks

We showed how to encode context-sensitivity in the 2-coloring model and constraint automata by adding fictitious nodes to every connector model, while both of these two formalisms are considered incapable of describing the behavior of context-dependent connectors. Our approach, constituted by the $\mathbb{M}$-transformation and the $\mathbb{L}$-transformation, enables the application of tools and algorithms devised for these simpler semantic models to context-dependent connectors. As an example, we demonstrated how Vereofy can model check context-sensitive connectors, which seemed impossible up to now.

*Future work* With respect to future work, we would like to investigate whether Reo's implementation can benefit from the results presented in this report. We speculate that algorithms for the computation of connector composition run faster on $\mathbb{M}$-transformed 2-colored $\eta$-connectors (or their corresponding $\alpha$-connectors) than on the untransformed 3-colored $\eta$-connectors, because of the simpler semantic model. Furthermore, we would like to extend our results that concern the correspondence between 2-coloring models and constraint automata (recall that in this report, we have formally established an equivalence in only one direction). Finally, we would like to investigate the relation between other formalisms for Reo that facilitate the proper modeling of context-dependent behavior (e.g., intentional automata and guarded automata).

*Acknowledgments* We are grateful to the Vereofy team for their support.

## References

AR03. Farhad Arbab and Jan Rutten. A coinductive calculus of component connectors. In Marin Wirsing, Dirk Pattinson, and Rolf Hennicker, editors, *Recent Trends in Algebraic Development Techniques*, volume 2755 of *LNCS*, pages 34–55. 2003. 55

Arb04. Farhad Arbab. Reo: A channel-based coordination model for component composition. *Mathematical Structures in Computer Science*, 14:329–366, 2004. 1, 2, 3, 22

BBKK09a. Christel Baier, Tobias Blechmann, Joachim Klein, and Sascha Klüppelholz. Formal verification for components and connectors. In Frank de Boer, Marcello Bonsangue, and Eric Madelaine, editors, *Formal Methods for Components and Objects*, volume 5751 of *Lecture Notes in Computer Science*, pages 82–101. 2009. 52

BBKK09b. Christel Baier, Tobias Blechmann, Joachim Klein, and Sascha Klüppelholz. A uniform framework for modeling and verifying components and connectors. In John Field and Vasco Vasconcelos, editors, *Coordination Models and Languages*, volume 5521 of *Lecture Notes in Computer Science*, pages 247–267. 2009. 52

BCS09. Marcello M. Bonsangue, Dave Clarke, and Alexandra Silva. Automata for context-dependent connectors. In John Field and Vasco Vasconcelos, editors, *Coordination Models and Languages*, volume 5521 of *LNCS*, pages 184–203. 2009. 56

BSAR06. Christel Baier, Marjan Sirjani, Farhad Arbab, and Jan Rutten. Modeling component connectors in Reo by constraint automata. *Science of Computer Programming*, 61(2):75–113, 2006. 1, 12, 13, 14, 15, 55

CCA07. Dave Clarke, David Costa, and Farhad Arbab. Connector colouring I: Synchronisation and context dependency. *Science of Computer Programming*, 66(3):205–225, 2007. 1, 2, 3, 6, 7, 9, 10, 45, 55

Cos10. David Costa. *Formal Models for Component Connectors*. PhD thesis, Vrije Universiteit Amsterdam, 2010. 3, 6, 7, 8, 9, 10, 11, 12, 37, 45, 53, 55

KC09. Christian Koehler and Dave Clarke. Decomposing port automata. In *Proceedings of the 2009 ACM Symposium on Applied Computing*, pages 1369–1373, 2009. 13

Kem09. Stephanie Kemper. SAT-based verification for timed component connectors. *Electronic Notes in Theoretical Computer Science*, 255:103–118, 2009. 56

KKdV10. Natallia Kokash, Christian Krause, and Erik P. de Vink. Verification of context-dependent channel-based service models. In Frank S. de Boer, Marcello M. Bonsangue, Stefan Hallerstede, and Michael Leuschel, editors, *Formal Methods for Components and Objects*, volume 6286 of *Lecture Notes in Computer Science*, pages 21–40. 2010. 56

Kra11. Christian Krause. *Reconfigurable Component Connectors*. PhD thesis, Universiteit Leiden, 2011. 16

KSA+08. Ramtin Khosravi, Marjan Sirjani, Nesa Asoudeh, Shaghayegh Sahebi, and Hamed Iravanchi. Modeling and analysis of Reo connectors using Alloy. In Doug Lea and Gianluigi Zavattaro, editors, *Coordination Models and Languages*, volume 5052 of *LNCS*, pages 169–183. 2008. 56

Mil89. Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989. 29, 47

MSA06. Mohammad Reza Mousavi, Marjan Sirjani, and Farhad Arbab. Formal semantics and analysis of component connectors in Reo. *Electronic Notes in Theoretical Computer Science*, 154(1):83–99, 2006. 56

PSAB11. Bahman Pourvatan, Marjan Sirjani, Farhad Arbab, and Marcello Bonsangue. Decomposition of constraint automata. In *Proceedings of the 7th International Workshop on Formal Aspects of Component Software (FACS 2010)*, To appear in LNCS. 2011. 2