



Reo Connectors and Components as Tagged Signal Models

Marjan Sirjani^{1,2} , Fatemeh Ghassemi³ , and Bahman Pourvatan^{2,4}

¹ School of Innovation, Design and Engineering, Mälardalen University,
Västerås, Sweden

`marjan.sirjani@mdh.se`

² School of Computer Science, Reykjavik University, Reykjavik, Iceland

`bahman@ru.is`

³ School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran

`fghassemi@ut.ac.ir`

⁴ LIACS, Leiden University, Leiden, The Netherlands

Abstract. Tagged Signal Model (TSM) is a denotational framework and a meta-model to study certain properties of models of computation. To study the behavior of Reo connectors in a closed system, we propose two denotational semantics for Reo using TSM. TSM is very similar to the coalgebraic model of Timed Data Streams (TDS), the first formal semantics and the basis for most of the other formal semantics of Reo. There is a direct mapping between the *time – data* pairs of TDS, and *tag – value* of TSM. This work shows how treating tags to be either totally or partially ordered has a direct consequence on the results. We looked into five primitive connectors of Reo in both these settings and discuss the determinacy of systems.

Foreword

Tagged Signal Model (TSM) and Timed Data Streams (TDS) are very similar mathematical models. We observed the extreme similarity and started this research with the goal to use the rich set of theorems and techniques built around the TSM framework for reasoning about different properties of Reo circuits. But the path we went through was not at all what we have expected. We have aimed for a discussion on determinacy of Reo connectors. We considered the five primitive Reo connectors as processes in TSM. We close the model, by adding source and sink processes to produce input for, and consume the output of each connector. The determinacy of such compositions as closed models depend on the source and sink processes. Moreover, we could not find any theorem on determinacy of models for the Rendezvous model of computation.

As we know Farhad’s love of Persian poetry we would like to cite the first Ghazal of Hafez here:

Ho! O Saki, pass around and offer the bowl:

For love appeared easy at first, but hardships have occurred.

We hope that Hafez forgives us for making an analogy between love and research.

1 Introduction

Development of concurrent systems has many challenges due to the well-known problems such as race conditions, synchronization of events, etc. Component-based development paradigm brings up a revolution in the software development. A system is made by composing off-the-self previously developed components. The glue code, by composing the components together, plays the important role of orchestration and defines how the components are coordinated to remedy the concurrency problems. Coordination languages have emerged for building the interaction protocols among the components in a system independent of the behavior of components. Reo was introduced as an exogenous coordination language to specify the glue code in a compositional way [1]. By composing ready-to-use components and Reo connectors, a system can be constructed. One of the challenges in the composition of Reo connectors is the interpretation of feedback loops. Feedback is a useful control mechanism, present in many coordination patterns, which may lead to non-deterministic behavior that it is not appealing.

Several behavioral semantics based on different formal classes have been appeared for Reo, namely, based on coalgebraic models, operational models [3], and graph-coloring [4]. The coalgebraic model of *timed data streams* (TDS) was the first model used to give semantics to Reo connectors [2]. It defines which and when data items flow through each node of a connector. To provide tools to support implementations or analysis of Reo connectors with formal techniques other semantics were developed. The TDS semantics define the behavior of connectors independent of interacting components (processes). In an ideal environment, there is an assumption that all the source and sink components (processes) are always willing to generate and consume data (and willing to Rendezvous). However such an assumption is not valid for all off-the-shelf components and hence, their composition may lead to unexpected behaviors. Here, we study the behavior of connectors and processes together as a composition.

The intuitive way of thinking about flows through a Reo connector resembles the way the behavior of processes is defined by *tagged signal models* (TSM) [6]. The denotational formalism of tagged signal model is a meta-model to study certain properties of models of computation in a unified framework. In this framework, a system is modeled by the composition of a set of processes. Each process is defined as a relation/function between signals which can be partitioned into input/output signals. Composition is treated as combining the output signals of one process to the inputs of some processes. Each signal is a set of tag-value pairs. By restricting processes to functions in TSM, feedback makes such systems self-referential. Mathematically, the notion of self-reference is tackled as a fixed point problem, as illustrated by the simple system in Fig. 1 in which the input and output signals are the same due to the feedback connection. Such a system has a well-defined behavior if F has a fixed point. This imposes constraints on the functions that are used to model systems. There are a set of well-defined theorems in the TSM framework to show when a model with feedback has behavior.

TSM defines precisely processes, signals, and events, and gives a framework for identifying the essential properties of discrete-event systems, dataflow,

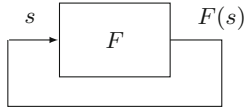


Fig. 1. A functional process in a feedback

rendezvous-based systems, Petri nets, and process networks. TSM is a meta-model, and TDS can be defined as a specific model, based on TSM, where TDS timestamps are a totally ordered set of tags in TSM. Furthermore, TSM provides sufficient means to describe connectors and interacting processes in a unified way as intended by Reo. In our approach, the effects of nodes are explicitly modeled by constraints in TSM. We use TSM framework to reason about the determinacy of a system composed of Reo connectors (and components). We will show that in the TSM for Reo, we can use the same totally ordered tag system as in the TDS, but most Reo circuits will be nondeterministic. We will also show the first steps towards a TSM for Reo with partially ordered set of tags, and how this model may be closer to the Rendezvous model of Reo.

In Sect. 2, we explain Reo and its Timed Data Stream semantics. We also explain the Tagged Signal Model, the possible structures of tag systems, and the determinacy of a system in this framework. In Sect. 3, we briefly point out to different semantics of Reo, and also to different places that TSM is used as the semantic framework for different models of computations. In Sect. 4, we show how to model Reo in the TSM framework with two different tag systems. Section 5 includes discussions and conclusions.

2 Preliminary Concepts

We first provide an overview of the syntax and semantics of the coordination language Reo, and then the meta-model of tagged signal framework.

2.1 Reo

Reo is a model for building component connectors in a compositional manner [1]. Each connector in Reo is, in turn, constructed compositionally out of simpler connectors, which are ultimately composed out of primitive channels.

A channel is a primitive communication medium with exactly two ends, each with its own unique identity. There are two types of channel ends: *source end* through which data enter and *sink end* through which data leave a channel. A channel must support a certain set of primitive operations, such as I/O, on its ends; beyond that, Reo places no restriction on the behavior of a channel.

A set of primitive Reo channels (together with their Timed Data Stream semantics) are shown in Table 1. Channels are connected to make a circuit. Connecting (or *joining*) channels is putting channel ends together in *nodes*. Thus, a set of channel ends is associated with a *node*. The semantics of a node depends

on its type. Based on the types of its coincident channel ends, a node can have one of three types. If all channel ends coincident on a node are exclusively source (or sink) channel ends, the node is called a source (respectively, sink) node. Otherwise, it is called a mixed node.

A component can write data items to a source node that it is connected to. The write operation succeeds only if all (source) channel ends coincident on the node accept the data item, in which case the data item is transparently written to every source end coincident on the node. A source node, thus, acts as a *replicator*. A component can obtain data items, by an input operation, from a sink node that it is connected to. A take operation succeeds only if at least one of the (sink) channel ends coincident on the node offers a suitable data item; if more than one coincident channel end offers suitable data items, one is selected nondeterministically. A sink node, thus, acts as a nondeterministic *merger*. A mixed node nondeterministically selects and takes a suitable data item offered by one of its coincident sink channel ends and replicates it into all of its coincident source channel ends.

Reo offers an open ended set of channels, but a set of primitive channels, shown in Fig. 1, are commonly used in Reo circuits. The behavior of every connector in Reo imposes a specific coordination pattern on the entities that perform normal (blocking) I/O operations through that connector, which itself is oblivious of those entities. This makes Reo a powerful *glue language* for compositional construction of connectors to combine component instances and Web services into a software system and exogenously orchestrate their mutual interactions.

2.2 Timed Data Stream

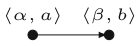
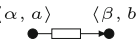
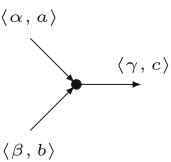
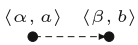
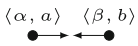
In [2], Arbab and Rutten introduce Timed Data Stream (TDS) models as the first formalization of the semantics of Reo connectors. Informally, a TDS model of a connector describes for each of its nodes which and when—in dense time—data items flow through this node. It does so by associating each node with a timed data stream. A timed data stream (α, a) consists of a *data stream* $\alpha \in Data^\omega$ and a monotonically increasing *time stream* $a \in R_{\geq}^\omega$ consisting of increasing positive real numbers including zero. The time stream a indicates for each data item α_n the moment a_n at which it is being input or output. By associating each node of a connector with its own TDS in a TDS tuple, a single execution of the connector is defined. To describe all possible executions of a connector, it has to be associated with a set of TDS tuples; we call such a set the TDS model of the connector. Usually, such a TDS model is defined as a predicate on TDSs that induces the set of admissible TDS tuples of a connector. (Enumerating all admissible TDS tuples of a connector becomes impossible because, not only each stream in a TDS itself is infinite, the set of admissible TDS tuples usually contains infinitely many elements.)

The Sync channel inputs the data elements in the stream α at time a , and outputs the date stream β at time b . All data elements that come in, come out in the same order, i.e., $\alpha = \beta$, and at the exact same time $a = b$. For a FIFO1 channel what comes in, comes out ($\alpha = \beta$), but at a later time ($a < b$).

Moreover, at any moment the next data item can only be input after the present data item has been output ($b < a'$) which means $b(n) < a(n + 1)$, for all $n \geq 0$. The connector **Merger** is a ternary relation with two input ends and one output end. This connector merges the two input data streams into a data stream on its output end. **Merger** handles the data element on one of its input ends first which comes out on its output end. The **LossySync** channel passes a data element on its input end instantaneously on as an output element, and continues with the remainder of the streams as before. If the output end is not ready for the rendezvous then the input element is discarded. The **SyncDrain** channel has two input ends, and the data elements in the stream α and β enter the two input ends of this channel simultaneously, i.e., $a = b$. There is no constraint on the data streams, the data elements enter and disappear.

Constraint automata can be viewed as acceptors for tuples of timed data streams that are observed at certain input/output ports A_1, \dots, A_n of components. The rough idea is that such an automaton observes the data occurring at A_1, \dots, A_n and either changes its state according to the observed data or rejects it if there is no corresponding transition in the automaton.

Table 1. Primitive connectors and their corresponding semantics as Timed Data Stream

Reo Connectors	Timed Data Stream
Synchronous Channel (Sync) 	$\langle \alpha, a \rangle \mapsto \langle \beta, b \rangle \equiv \alpha = \beta \wedge a = b$
FIFO with one buffer (FIFO1) 	$\langle \alpha, a \rangle \dashrightarrow \langle \beta, b \rangle \equiv \alpha = \beta \wedge a < b < a'$
Merger Connector (Merger) 	$M(\langle \alpha, a \rangle, \langle \beta, b \rangle, \langle \gamma, c \rangle) \equiv a(0) \neq b(0) \wedge$ if $a(0) < b(0) \quad \alpha(0) = \gamma(0) \wedge a(0) = c(0) \wedge M(\langle \alpha', a' \rangle, \langle \beta, b \rangle, \langle \gamma', c' \rangle)$ if $b(0) < a(0) \quad \beta(0) = \gamma(0) \wedge b(0) = c(0) \wedge M(\langle \alpha, a \rangle, \langle \beta', b' \rangle, \langle \gamma', c' \rangle)$
Lossy Synchronous Channel (LossySync) 	$\langle \alpha, a \rangle \dashrightarrow \langle \beta, b \rangle \equiv a(0) \leq b(0) \wedge$ if $a(0) = b(0) \quad \alpha(0) = \beta(0) \wedge \langle \alpha', a' \rangle \dashrightarrow \langle \beta', b' \rangle$ if $a(0) < b(0) \quad \langle \alpha', a' \rangle \dashrightarrow \langle \beta, b \rangle$
Synchronous Drain (SyncDrain) 	$\langle \alpha, a \rangle \rightarrow \leftarrow \langle \beta, b \rangle \equiv a = b$

2.3 Tagged Signal Model

Let T and V denote the set of tags and values, ranged over by t and v respectively. An event e is a pair of a tag and a value, where its tag may denote the time that the event has occurred while its value may represent the operand/result of a computation. A signal s is a set of events, and the set of all signals S is defined by the powerset $\wp(T \times V)$. A tuple \mathbf{s} of N signals written by $\mathbf{s} = (s_1, \dots, s_N)$, is used to model the behavior of a process; and a process which is a unit of computation is a set of behaviors. We use the position i in the tuple to denote the signal s_i . The set of all such tuples is denoted by S^N . The empty signal is denoted by $\lambda \in S$ while the tuple of empty signals by $\Lambda \in S^N$.

In this framework, a system is modeled by a composition of a set of processes. A process P is described in its general term as a subset of S^N , called its *sort*. A particular $\mathbf{s} \in S^N$ is called a *behavior* of P if $\mathbf{s} \in P$. For $N \geq 2$, a process can also be interpreted as a relation/function between the N signals in \mathbf{s} which can be partitioned into input/output signals.

A composition of processes is simply defined by the intersection of the behaviors of the processes. In order to be able to compose processes, all processes have to be modeled using the same sort. This is one of the more subtle aspects of TSM. This means that the set of behaviors of a process includes signals that are neither inputs nor outputs to the process, that the process has nothing to do with. Every possible valuation of such signals can be found in the behaviors of the process. When composing processes by intersection, a process has no effect on signals that are irrelevant to it because all possible valuations of those signals are legitimate behaviors of the process, and when we form set intersection, these irrelevant signals are not constrained by the process in any way. So, for composition, processes are defined as a subset of the same sort by augmenting their tuples using cross product.

Interaction is defined by the particularly simple process $C \subset S^N$, called *connection*, where two (or more) of the signals in the N -tuple are constrained to be identical. Connections are useful to couple the behaviors of other processes. For example, the connection $C_{i,j} = \{s \in S^N \mid s_i = s_j\}$, intuitively models that the signals s_i and s_j are connected together. To hide some signals, the projection operator $\pi_I(\mathbf{s})$ is used which maps $s = (s_1, \dots, s_N)$ to $(s_{i_1}, \dots, s_{i_m})$ where $I = \{i_1, \dots, i_m\}$ is an ordered set of indexes in the range $1 \leq i \leq N$. For instance the composite process in the right-side of Fig. 3 is defined by $\pi_{1,4}((P_1 \times S^2) \cap (S^2 \times P_2) \cap C_{2,3})$ which can be simply denoted by $\pi_{1,4}((P_1 \times P_2) \cap C_{2,3})$.

A process P is functional with respect to the index sets I and O for m and n input and output signals respectively, if for every $\mathbf{s} \in P$ and $\mathbf{s}' \in P$, $\pi_I(\mathbf{s}) = \pi_I(\mathbf{s}')$ implies $\pi_O(\mathbf{s}) = \pi_O(\mathbf{s}')$. Therefore, for the functional process P with respect to (I, O) , a single-valued mapping $F : S^m \rightarrow S^n$ can be defined such that for all $\mathbf{s} \in P$, $\pi_O(\mathbf{s}) = F(\pi_I(\mathbf{s}))$. Note that a process may be functional with respect to more than one pair of index set (I, O) . This property is preserved by the composition of functional processes as long as no feedback loop is involved.

Tag Systems. Intuitively tags are used to model time, precedence relations, synchronizations points, etc. The central role of a tag system is to establish ordering among events. The structure of a tag system distinguishes various concurrent models of computation, classified into *timed* and *untimed*. The former characterizes systems in which the order of events is deterministically defined relative to some physical or logical clock. Therefore, timed models of computation are characterized by a totally ordered set of tags while untimed ones by a partially ordered set of tags. In a timed model, the order of all events is clear and all tags are comparable, while in an untimed model, a subset of events can be ordered.

Determinacy. Many processes (not necessarily functional) have the notion of inputs which characterize events or signals that are defined outside the process. Formally, an input to the process $P \subseteq S^N$ is an externally imposed constraint $A \subseteq S^N$ such that $A \cap P$ is the total set of acceptable behaviors. The behavior of a process for a set of possible inputs, denoted by $B \subseteq \wp(S^N)$, can be defined by (P, B) .

A process (P, B) is called *closed* if $B = \{S^N\}$, a set with only one element, $A = S^N$. Since $A \cap P = P$, no input constraints are imposed on a closed process. A process and its possible inputs is *open* if it is not closed.

A process is called deterministic “if for any input $A \in B$ it has exactly one behavior or exactly no behavior; i.e. $|A \cap P| = 1$ or $|A \cap P| = 0$, where $|X|$ is the size of the set X .” Otherwise, it is called nondeterministic. Consequently, a closed process P is deterministic if $|P| = 1$ or $|P| = 0$ (because $B = \{S^N\}$ and $|S^N \cap P| = 1$ or $|S^N \cap P| = 0$).

A functional process with respect to (I, O) is obviously deterministic if I and O together contain all the indexes in $1 \leq i \leq N$.

3 Related Work

In recent years, many formalisms for describing the behavior of Reo connectors have emerged. Jongmans and Arbab provided an overview of thirty different semantic formalisms for Reo in [4]. These models include coalgebraic models, operational models, and models based on graph-coloring. In [4], the authors also investigate in more detail the expressiveness of constraint automata and coloring models. We encourage the interested reader to [4] for more detailed information.

The first formal semantics proposed for Reo is Timed Data Streams proposed in [2]. In [2], like in most other semantics proposed for Reo, the focus is on the set of connectors and their composition, and the behavior of components is abstracted away and substituted by (sometimes implicit) assumptions. Tagged Signal Model is introduced in [6] as a denotational framework for comparing models of computation. It is a generalization of the Signals and Systems approach to system modeling and specification [7]. Using Tagged Signal Model, one can give structure to the sets of signals, give structure to the functional processes, and develop static analysis techniques. Moreover, we can compare certain properties

of the models of computation, such as their notion of synchrony, and define formal relations among signals and process behaviors.

The similarities between tagged signals and timed data streams motivated us to look into Reo semantics using TSM framework. The goal is to use the established theory around Tagged Signal Model for reasoning about different properties of Reo circuits. This paper is the first attempt in moving towards this goal. Although TSM and TDS are very similar in their structure, our work in this paper shows subtle problems that need to be solved to be able to use the fixed point theorems established for TSM in the context of Reo connectors. TSM is mainly used to reason about Kahn Process Networks and Discrete Event model of computation. Reo is using a Rendezvous model of computation, and there is no TSM proposed for such models. Comparing to the TDS model, the TSM framework will add the ability to also model processes rather than just connectors. We will show that in the TSM for Reo, we can use the same totally ordered tag system as in the TDS, but most Reo circuits will be nondeterministic. We will also show the first steps towards a TSM for Reo with partially ordered set of tags, and how this model may be closer to the Rendezvous model of Reo.

4 Reo Connectors as Tagged Signal Models

To study the notions of concurrency, determinacy and synchronization of Reo connectors, we define how these properties can be captured within the tagged signal model. We provide a denotational semantics for Reo depending on how the tag system is structured.

In the first setting, the tag system is considered to be totally ordered and is the set of non-negative real numbers. This structure is inspired from the semantics of Reo as Timed Data Streams in [2]. As there is a global view of time among the components, it can be considered that all components and connectors are local. This setting can be extended to provide a suitable model for a discrete-event simulator of Reo connectors.

In the second setting, the tag system is considered to be partially ordered. In this setting, the components interacting with connectors and the nodes of the connectors are considered to be distributed, and hence such a tag system reflects the inherent problem in having a consistent time view in the implementation of distributed systems. In this setting, we may receive signals with incomparable tags at each channel end of a primitive connector (primitive channels or the merger). So, keeping the Reo node behavior similar to the first setting, our proposed TSM in the second setting comes short in defining semantics for all the primitive connectors.

The model of computation in Reo nodes is rendezvous, and Reo nodes take care of signals received from different (sink) channel ends with (possibly) incomparable partially ordered tags and dispatch them accordingly to the (source) channel ends. This way, we assume comparable tags for input and output signals of each primitive connector (similar to [2]). To be able to capture the semantics of Reo properly, in our future work we intend to adopt a partially ordered tag

system, keep the semantics of each primitive connector aligned with their TDS definitions in [2], and add a more elaborated semantics for Reo nodes as special TSM connectors which compose channel ends and enforce the rendezvous model. Such an adoption may make it possible to use the well-established theorems around Network Process [5] to reason about the determinacy systems composed of Reo connector with feedback loops. Establishing all the details of this third model is left as our future work, and in this paper we show how we moved towards this semantics.

In all the models, each Reo channel can be viewed as a process, and its input/output streams of data in to and out of channel ends are viewed as signals.

4.1 A Totally Ordered Tag Model

When the tag set is totally ordered, for any two distinct tags $t, t' \in T$, either $t < t'$ or $t' < t$. We consider T to be the set of non-negative real numbers. We say $e_1 < e_2$ when $e_1 = (t_1, v)$ and $e_2 = (t_2, v)$ for some $v \in V$ where $t_1 < t_2$. Let $T(e)$ denote the tag of the event e , and let $T(s)$ denotes the set of tags of all events of the signal s .

We express the semantics of the five basic connectors as shown in Table 2. We assume that signals s_1 and s_2 are totally ordered:

$$\begin{aligned} s_1 &= \{e_i, i \in \mathbb{N}\}, \forall i, j \cdot i < j \Rightarrow T(e_i) < T(e_j) \\ s_2 &= \{e'_i, i \in \mathbb{N}\}, \forall i, j \cdot i < j \Rightarrow T(e'_i) < T(e'_j) \end{aligned}$$

In this setting, the process of **Sync** is functional which can be defined by the identity function. However, the **FIFO1** connector is not a functional process. The asynchronous behavior of **FIFO1** makes it non-deterministic as for any $(t, v) \in s_1$, it can be delivered to s_2 at any time $t' > t$.

Here, **Merger** can be seen as a partial function process. The constraint $T(s_1) \cap T(s_2) = \emptyset$ expresses that its behavior is not defined when the tags of two events at its input signals are equal. The semantics of **LossySync** is defined by a relation due to its non-deterministic behavior in losing events. The process of **SyncDrain** is defined as a partial function.

Two connectors can be composed through a node which contains the channel ends of the both connectors. Such a node imposes constraints on the signals representing the channel ends being contained by the node. For instance, the behavior of the mixed node in Fig. 2 is defined by the constraint $C'_{1,\{2,\dots,n\}}$ which faithfully models the effect of nodes in Reo:

$$C'_{1,\{2,\dots,n\}} = \{(s_1, \dots, s_n) \mid \forall 2 < j \leq n \cdot (s_2 = s_j) \wedge \forall e_i \in s_1, e'_i \in s_2 \cdot (T(e_i) \leq T(e'_i))\}.$$

The semantics of a system is derived by the intersection of the behaviors of it's constituent processes.

Example 1. Consider a connector composed of two **Sync** channels connected in sequence, as shown in Fig. 3. The behavior of the composite connector is defined

Table 2. The primitive connectors Sync, FIFO1, Merger, LossySync, and SyncDrain and their corresponding semantics with a totally ordered tag model. Note that $s_1 = \{e_i, i \in \mathbb{N}\}$ where $\forall i, j \cdot i < j \Rightarrow T(e_i) < T(e_j)$, and $s_2 = \{e'_i, i \in \mathbb{N}\}$ where $\forall i, j \cdot i < j \Rightarrow T(e'_i) < T(e'_j)$, and we say $e_1 < e_2$ when $e_1 = (t_1, v)$ and $e_2 = (t_2, v)$ for some $v \in V$ where $t_1 < t_2$.

Reo Connector	Process	Behavior
		$P = \{(s_1, s_2) \mid s_1 = s_2\}$
		$P = \{(s_1, s_2) \mid \forall k \in \mathbb{N} \cdot (e_k < e'_k < e_{k+1})\}$
		$P = \{(s_1, s_2, s_3) \mid s_3 = s_1 \cup s_2 \wedge T(s_1) \cap T(s_2) = \emptyset\}$
		$P = \{(s_1, s_2) \mid s_2 \subseteq s_1\}$
		$P = \{(s_1, s_2) \mid T(s_1) = T(s_2)\}$

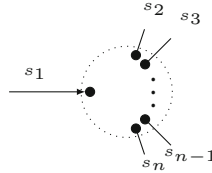


Fig. 2. A mixed node in Reo, including one source channel end (related to s_1), and multiple sink channel ends (related to s_2 to s_n)

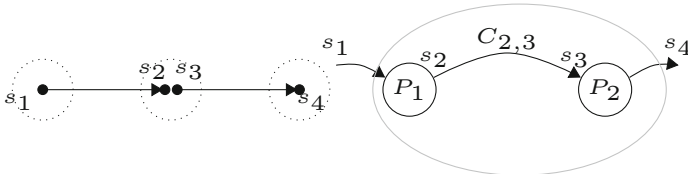


Fig. 3. Two Sync channel in sequence and its corresponding composite process

by the intersection of two functional processes P_1 and P_2 and the connection $C_{2,3}$. Therefore, Their composition is functional, and hence, since their composition is acyclic, it is deterministic. This can be validated by computing the behavior of the connector:

$$\pi_{1,4}((P_1 \times P_2) \cap C_{2,3}) = \{(s_1, s_4) \mid s_1 = s_4\}.$$

Example 2. Consider a system composed of a Merger connector and three components C_1 , C_2 , and C_3 , as shown in Fig. 4. Assume that C_1 and C_2 are always willing to generate data while C_3 is always willing to consume data, defined by processes P_1 , P_2 , and P_3 , respectively:

$$P_i = \{(s_1, s_2, s_3, s_4, s_5, s_6) \mid T(s_i) = R_{\geq}^{\omega}\}, i \in \{1, 2, 3\}.$$

The behavior of the composite closed system is defined by the intersection of all processes, the corresponding process of Merger, modeled by M and the constraints $C'_{1,\{4\}}$, $C'_{2,\{5\}}$, and $C'_{6,\{3\}}$. The behavior of the composite system is:

$$P_1 \cap P_2 \cap P_3 \cap M \cap C'_{1,\{4\}} \cap C'_{2,\{5\}} \cap C'_{6,\{3\}}$$

where M is the augmented behavior of Merger:

$$M = \{(s_1, s_2, s_3, s_4, s_5, s_6) \mid s_6 = s_4 \cup s_5 \wedge T(s_4) \cap T(s_5) = \emptyset\}.$$

The composition yields an uncountably infinite set and hence, defines a non-deterministic system regarding the definition of determinacy in Sect. 2.3. Replacing C_1 and C_2 by components with only one behavior will make the system deterministic.

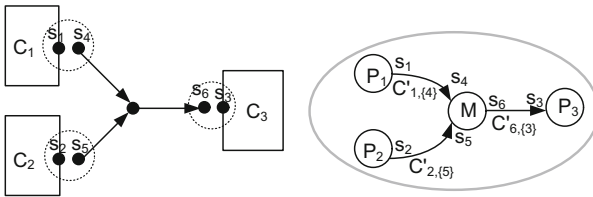


Fig. 4. A closed system composed of a Merger

Example 3. We change the system of Example 2 by connecting the inputs of the merger to a SyncDrain channel, as shown in Fig. 5. We assume again that C_1 (similarly C_2) and C_3 are always willing to generate and consume data respectively. The behavior of the composite system is:

$$P_1 \cap P_2 \cap P_3 \cap D \cap M \cap C'_{1,\{4,7\}} \cap C'_{2,\{5,8\}} \cap C'_{6,\{3\}}$$

where M and D are the augmented behaviors of the Merger and SyncDrain channels, respectively:

$$M = \{(s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8) \mid s_6 = s_4 \cup s_5 \wedge T(s_4) \cap T(s_5) = \emptyset\}$$

$$D = \{(s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8) \mid T(s_7) = T(s_8)\}.$$

The composition yields an empty set and hence, results in a deterministic system.

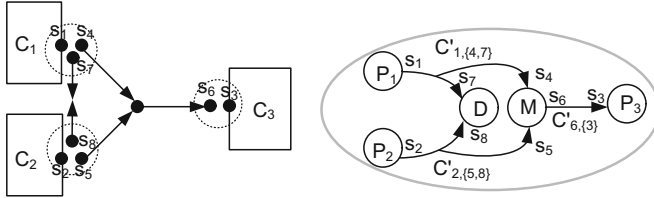


Fig. 5. A closed system composed of a Merger and a SyncDrain

4.2 A Partially Ordered Tag Model


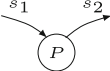

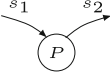

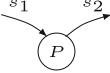
As explained before, a partially ordered tag model seems more suitable in a distributed setting for Reo connectors. Since different components have no consistent view of time, only events generated by the same primitive connector (component) at its sink channel ends (output ports of the component), and events coming into the primitive connector from the same source channel end (each input port of the component) are totally-ordered. When two sequences of events (generated by different channels/components) arrive at the channel ends of a Merger or SyncDrain, their tags cannot be compared. Therefore, the behavior of Merger and SyncDrain cannot be defined in this setting without further elaborations.

The semantics of Sync, FIFO1, and LossySync channels, provided in Table 3, are similar to the ones given in the timed model of computation (totally ordered tags). The Sync channel passes the events without manipulating their tags and values. The behavior of FIFO1 ensures that the tag of each outgoing event is greater than its corresponding incoming event but less than the next incoming event. The behavior of LossySync channel shows that if an event passes through the channel, its tag and value will not be changed, but passing an event is not guaranteed. In contrast to the Sync channel, due to its nondeterministic behavior in losing an event its process is not functional.

The behavior of a node is exactly modeled as in the totally ordered tags setting. It should be noted that we assumed that a node and its constituent ends are located at the same place, and hence their tags are comparable.

Providing a model for Merger, and SyncDrain channel needs more elaboration, and we leave it to our future work.

Table 3. The primitive connectors Sync, FIFO1, and LossySync channels and their corresponding semantics with a partially ordered tag model. Note that $s_1 = \{e_i, i \in \mathbb{N}\}$ where $\forall i, j \cdot i < j \Rightarrow T(e_i) < T(e_j)$, and $s_2 = \{e'_i, i \in \mathbb{N}\}$ where $\forall i, j \cdot i < j \Rightarrow T(e'_i) < T(e'_j)$, and we say $e_1 < e_2$ when $e_1 = (t_1, v)$ and $e_2 = (t_2, v)$ for some $v \in V$ where $t_1 < t_2$.

Reo Connector	Process	Behavior
		$P = \{(s_1, s_2) \mid s_1 = s_2\}$
		$P = \{(s_1, s_2) \mid \forall k \in \mathbb{N} \cdot (e_k < e'_k < e_{k+1})\}$
		$P = \{(s_1, s_2) \mid s_2 \subseteq s_1\}$

5 Discussion, Conclusion and Future Work

We observed the similarity between Tagged Signal Model presented in [6] and Timed Data Stream presented in [2]. Different techniques are established based on Tagged Signal Model to understand the behavior of a model of computation better, and to reason about the determinacy of the composition of processes at the syntactic level using fixed point theory. Our main motivation was to discuss the possible nondeterministic behaviors of different Reo connectors specially when a feedback loop is formed in the composition, by exploiting TSM framework and its results on systems with feedback loops. We provided two denotational semantics for Reo connectors in the two timed and untimed models of computation, based on totally and partially ordered sets of tags respectively.

Moving towards a partially ordered set of tags, we may stick to the way the primitive connectors are modeled in [2], but we need to show how we compare different tags coming from different sources where necessary, or show how and where we can resolve this comparison. We need to consider the rendezvous that is happening within Reo nodes to model the change of tags and propagation of change of tags on the upstream signals coming in from other processes, and downstream signals going to other processes. The details for this model is left as the future work.

Acknowledgment. We would like to thank Professor Edward Lee for his very useful comments and discussions.

References

1. Arbab, F.: Reo: a channel-based coordination model for component composition. *Math. Struct. Comput. Sci.* **14**(3), 329–366 (2004)
2. Arbab, F., Rutten, J.J.M.M.: A coinductive calculus of component connectors. In: Wirsing, M., Pattinson, D., Hennicker, R. (eds.) *WADT 2002*. LNCS, vol. 2755, pp. 34–55. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-40020-2_2
3. Baier, C., Sirjani, M., Arbab, F., Rutten, J.J.M.M.: Modeling component connectors in Reo by constraint automata. *Sci. Comput. Program.* **61**(2), 75–113 (2006)
4. Jongmans, T.Q., Arbab, F.: Overview of thirty semantic formalisms for Reo. *Sci. Ann. Comput. Sci.* **22**(1), 201–251 (2012)
5. Lee, E.A.: *Concurrent Models of Computation - An Actor-Oriented Approach (Draft)* (2011)
6. Lee, E.A., Sangiovanni-Vincentelli, A.L.: A framework for comparing models of computation. *IEEE Trans. CAD Integr. Circuits Syst.* **17**(12), 1217–1229 (1998)
7. Liu, X.: *Semantic foundation of the tagged signal model*. Ph.D. thesis, EECS Department, University of California, Berkeley (2005)