# Amending Choreographies

Ivan Lanese
Focus Team, University of Bologna/INRIA, Italy
lanese@cs.unibo.it

Fabrizio Montesi
IT University of Copenhagen, Denmark
fmontesi@itu.dk

Gianluigi Zavattaro
Focus Team, University of Bologna/INRIA, Italy
zavattar@cs.unibo.it

Choreographies are global descriptions of system behaviors, from which the local behavior of each endpoint entity can be obtained automatically through projection. To guarantee that its projection is correct, i.e. it has the same behaviors of the original choreography, a choreography usually has to respect some coherency conditions. This restricts the set of choreographies that can be projected.

In this paper, we present a transformation for amending choreographies that do not respect common syntactic conditions for projection correctness. Specifically, our transformation automatically reduces the amount of concurrency, and it infers and adds hidden communications that make the resulting choreography respect the desired conditions, while preserving its behavior.

## 1 Introduction

Choreography-based programming is a powerful paradigm where the programmer defines the communication behavior of a system from a global viewpoint, instead of separately specifying the behavior of each endpoint entity. Then, the local behavior of each endpoint can be automatically generated through a notion of *projection* [3, 6, 7, 1, 4]. A projection procedure is usually a homomorphism from choreographies to endpoint code. However, projections of some choreographies may lead to undesirable behavior. To characterize the class of choreographies that can be correctly projected some syntactic conditions have been put forward. Consider the following choreography $C$:

$$C = \mathtt{a} \to \mathtt{b}\,{:}\,o_1;\ \mathtt{c} \to \mathtt{d}\,{:}\,o_2$$

Here, $\mathtt{a}$, $\mathtt{b}$, $\mathtt{c}$ and $\mathtt{d}$ are *participants* and $o_1$ and $o_2$ are *operations* (labels for communications). $C$ specifies a system where $\mathtt{a}$ sends to $\mathtt{b}$ a message on operation $o_1$ ($\mathtt{a} \to \mathtt{b}\,{:}\,o_1$), and then (; is sequential composition) $\mathtt{c}$ sends to $\mathtt{d}$ a message on operation $o_2$ ($\mathtt{c} \to \mathtt{d}\,{:}\,o_2$). We can naturally project $C$ onto a system $S$ composed of CCS-like processes annotated with roles [7]:

$$S = [\overline{o_1}]_{\mathtt{a}} \parallel [o_1]_{\mathtt{b}} \parallel [\overline{o_2}]_{\mathtt{c}} \parallel [o_2]_{\mathtt{d}}$$

Here, $[\cdot]_{\mathtt{a}}$ specifies the behavior of participant $\mathtt{a}$, $\parallel$ is parallel composition, $\overline{o_1}$ denotes an output on operation $o_1$, and $o_1$ the corresponding input. Unfortunately, the projected system $S$ does not implement $C$ correctly. Indeed, $\mathtt{c}$ could send its message on $o_2$ to $\mathtt{d}$ before the communication on $o_1$ between $\mathtt{a}$ and $\mathtt{b}$ has occurred. This is in contrast with the intuitive meaning of the sequential composition operator ; in the choreography, which explicitly models sequentiality between the two communications. In [7], we provide syntactic conditions on choreographies for ensuring that projection will behave correctly. Similar conditions are used in other works (e.g., [3, 6]). Such conditions would reject choreography $C$

A. Ravara and J. Silva (Eds.): 9th Workshop on
Automated Specification and Verification of Web Systems (WWV'13)
EPTCS 123, 2013, pp. 34–48, doi:10.4204/EPTCS.123.5

above, by recognizing that it is not *connected*, i.e., the order of communications specified in $C$ cannot be enforced by the projected roles.

In this paper, we present a procedure for automatically transforming, or *amending*, a choreography that is not connected into a behaviorally equivalent one that is connected. Our transformation acts in two ways. In most of the cases, it automatically infers and adds some extra hidden communications. In a few cases instead the problem cannot be solved by adding communications, and the amount of concurrency in the system has to be decreased. For example, we can transform $C$ in the connected choreography $C'$ below:

$$C' = \mathtt{a} \rightarrow \mathtt{b}\!:\!o_1;\ \mathtt{b} \rightarrow \mathtt{e}\!:\!o_3^*;\ \mathtt{e} \rightarrow \mathtt{c}\!:\!o_4^*;\ \mathtt{c} \rightarrow \mathtt{d}\!:\!o_2$$

Here, we have added an extra role $\mathtt{e}$ that interacts with $\mathtt{b}$ and $\mathtt{c}$ to ensure the sequentiality of their respective communications. This is the first kind of transformation discussed above. We refer to Example 2 in Section 3.4 for an example of the second kind.

Our transformation brings several benefits. First, designers could use choreographies by defining only the desired behavior, leaving to our transformation the burden of filling in the necessary details on how the specified orderings should be enforced. Moreover, our solution does not change the standard definition of projection, since we operate only at the level of choreographies. Hence, we retain simplicity in the definition of projection. Finally, our transformation offers an automatic way of ensuring correctness when composing different choreographies developed separately into a single one.

**Structure of the paper:** Section 2 introduces choreographies and their semantics. Section 3 contains the main contribution of the paper, namely the description and the proof of correctness of the amending techniques. Section 4 applies the developed theory to a well-known example, the two-buyers protocol. Finally, Section 5 discusses related work and future directions. Appendix A and Appendix B describe projected systems and the projection operation, respectively.

## 2 Choreography Semantics

We introduce here our language for modeling choreographies. The set of participants in a choreography, called *roles*, is ranged over by $\mathtt{a}$, $\mathtt{b}$, $\mathtt{c}$, .... We also consider two kinds of *operations* for communications: *public operations*, ranged over by $o$, which represent observable activities of the system, and *private operations*, ranged over by $o^*$, used for internal synchronization. We use $o^?$ to range over both public and private operations.

Choreographies, ranged over by $C, C', \ldots$, are defined as follows:

$$C ::= \mathtt{a} \rightarrow \mathtt{b}\!:\!o^? \mid \mathbf{1} \mid \mathbf{0} \mid C;C' \mid C \parallel C' \mid C + C'$$

An interaction $\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?$ means that role $\mathtt{a}$ sends a message on operation $o^?$ to role $\mathtt{b}$ (we assume that $\mathtt{a} \neq \mathtt{b}$). Besides, there are the empty choreography $\mathbf{1}$, the deadlocked choreography $\mathbf{0}$, sequential and parallel composition of choreographies, and nondeterministic choice between choreographies. Deadlocked choreography $\mathbf{0}$ is only used at runtime, and it is not part of the user syntax.

The semantics of choreographies is the smallest labeled transition system (LTS) closed under the rules in Figure 1. Symmetric rules for parallel composition and choice have been omitted. We use $\sigma$ to range over labels. We have two kinds of labels: label $\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?$ denotes the execution of an interaction $\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?$ while label $\surd$ represents the termination of the choreography. Rule INTERACTION executes an interaction. Rule END terminates an empty choreography. Rule SEQUENCE executes a step in the

$$\text{(INTERACTION)}\quad \mathtt{a} \to \mathtt{b}{:}o^{?} \xrightarrow{\mathtt{a}\to\mathtt{b}:o^{?}} \mathbf{1} \qquad \text{(END)}\quad \mathbf{1} \xrightarrow{\surd} \mathbf{0} \qquad \text{(SEQUENCE)}\quad \frac{C \xrightarrow{\sigma} C' \quad \sigma \neq \surd}{C;C'' \xrightarrow{\sigma} C';C''} \qquad \text{(PARALLEL)}\quad \frac{C \xrightarrow{\sigma} C' \quad \sigma \neq \surd}{C \parallel C'' \xrightarrow{\sigma} C' \parallel C''}$$

$$\text{(CHOICE)}\quad \frac{C \xrightarrow{\sigma} C'}{C + C'' \xrightarrow{\sigma} C'} \qquad \text{(SEQ-END)}\quad \frac{C_1 \xrightarrow{\surd} C_1' \quad C_2 \xrightarrow{\sigma} C_2'}{C_1;C_2 \xrightarrow{\sigma} C_2'} \qquad \text{(PAR-END)}\quad \frac{C_1 \xrightarrow{\surd} C_1' \quad C_2 \xrightarrow{\surd} C_2'}{C_1 \parallel C_2 \xrightarrow{\surd} C_1' \parallel C_2'}$$

Figure 1: Choreographies, semantics (symmetric rules omitted).

first component of a sequential composition. Rule PARALLEL executes an interaction from a component of a parallel composition, while rule CHOICE starts the execution of an alternative in a nondeterministic choice. Rule SEQ-END acknowledges the termination of the first component of a sequential composition, starting the second component. Rule PAR-END synchronizes the termination of two parallel components.

We use the semantics to build some notions of traces for choreographies, which will be used later on for stating our results.

**Definition 1** (Choreography traces)**.** *A (strong maximal) trace of a choreography $C_1$ is a sequence of labels $\tilde{\sigma} = \sigma_1, \ldots, \sigma_n$ such that there is a sequence of transitions $C_1 \xrightarrow{\sigma_1} \ldots \xrightarrow{\sigma_n} C_{n+1}$ and that $C_{n+1}$ has no outgoing transitions.*

*A weak trace of a choreography $C$ is a sequence of labels $\tilde{\sigma}$ obtained by removing all the labels of the form $\mathtt{a} \to \mathtt{b}{:}o^*$ (corresponding to private interactions) from a strong trace of $C$.*

*Two choreographies $C$ and $C'$ are (weak) trace equivalent iff they have the same set of (weak) traces.*

## 3 Amending Choreographies

In this section we consider three different connectedness properties, and we show how to enforce each one of them, preserving the set of weak traces of the choreography. We refer to [8] (a technical report extending [7]) for the description of why these properties guarantee projectability. Actually, in [7], different notions of projectability are considered according to whether the semantic model is synchronous or asynchronous, and on which behavioral relation between a choreography and its projection is required. The conditions presented and enforced below are correct and ensure projectability according to all the notions considered in [7].

All the conditions below are stated at the level of choreographies. Nevertheless, definitions of projection, projected system and of its semantics may help in understanding why they are needed. We collect these definitions in the Appendix.

We discuss the connectedness conditions in increasing order of difficulty to help the understanding. Then, in Section 3.4, we combine them to make a general choreography connected.

### 3.1 Connecting Sequences

*Connectedness for sequence* ensures that, in a sequential composition $C;C'$, the choreography $C'$ starts its execution only after $C$ terminates. To formalize this property, we first need to define auxiliary functions transI and transF, which compute respectively the sets of initial and final interactions in a choreography:

$$\begin{array}{l}
\text{transI}(\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?) = \text{transF}(\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?) = \{\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?\}\\
\text{transI}(\mathbf{1}) = \text{transI}(\mathbf{0}) = \text{transF}(\mathbf{1}) = \text{transF}(\mathbf{0}) = \emptyset\\
\text{transI}(C \parallel C') = \text{transI}(C + C') = \text{transI}(C) \cup \text{transI}(C')\\
\text{transF}(C \parallel C') = \text{transF}(C + C') = \text{transF}(C) \cup \text{transF}(C')\\
\text{transI}(C;C') = \text{transI}(C) \cup \text{transI}(C') \text{ if } \exists C'' \text{ such that } C \xrightarrow{\surd} C'',\ \ \text{transI}(C) \text{ otherwise}\\
\text{transF}(C;C') = \text{transF}(C) \cup \text{transF}(C') \text{ if } \exists C'' \text{ such that } C' \xrightarrow{\surd} C'',\ \ \text{transF}(C') \text{ otherwise}
\end{array}$$

Intuitively, connectedness for sequence can be ensured by guaranteeing that a role waits for termination of all the interactions in $C$ and only then starts the execution of the interactions in $C'$. We can now formally state this property.

**Definition 2** (Connectedness for sequence). *A choreography $C$ is* connected for sequence *iff each subterm of the form $C';C''$ satisfies* $\forall \mathtt{a} \rightarrow \mathtt{b}\!:\!o_1^? \in \text{transF}(C'), \forall \mathtt{c} \rightarrow \mathtt{d}\!:\!o_2^? \in \text{transI}(C'').\mathtt{b} = \mathtt{c}$.

Our transformation reconfigures the subterms $C';C''$ failing to meet the condition. Take one such term $C';C''$. Choose a fresh role $\mathtt{e}$. Consider all the interactions $\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?$ contributing to $\text{transF}(C')$ in the term. For each of them choose a fresh operation $o_f^*$ and replace $\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?$ with $\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?;\ \mathtt{b} \rightarrow \mathtt{e}\!:\!o_f^*$. Similarly, for each interaction $\mathtt{c} \rightarrow \mathtt{d}\!:\!o^?$ contributing to $\text{transI}(C'')$ choose a fresh operation $o_f^*$ and replace $\mathtt{c} \rightarrow \mathtt{d}\!:\!o^?$ with $\mathtt{e} \rightarrow \mathtt{c}\!:\!o_f^*;\ \mathtt{c} \rightarrow \mathtt{d}\!:\!o^?$. The essential idea is that role $\mathtt{e}$ waits for all the threads in $C'$ to terminate before starting threads in $C''$.

**Proposition 1.** *Given a choreography $C$ we can derive using the pattern above a choreography $C'$ which is connected for sequence such that $C$ and $C'$ are weak trace equivalent.*

*Proof.* We have to apply the pattern to all the subterms failing to satisfy the condition. We start from smaller subterms, and then move to larger ones. We have to show that at the end all the subterms satisfy the condition. The new subterms introduced by the transformation have the form $\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?;\mathtt{b} \rightarrow \mathtt{e}\!:\!o^*$ and $\mathtt{e} \rightarrow \mathtt{c}\!:\!o^*;\mathtt{c} \rightarrow \mathtt{d}\!:\!o^?$, thus they satisfy the condition by construction. Let us consider a subterm $D';D''$ obtained by transforming a subterm $E';E''$ which already satisfied the condition (but whose subterms may have been transformed). It is easy to check that $\text{transF}(D') = \text{transF}(E')$ and $\text{transI}(D'') = \text{transI}(E'')$, thus the term is still connected for sequence. For subterms obtained by transforming subterms that did not satisfy the condition, the thesis holds by construction.

Weak trace equivalence is ensured since only interactions on private operations, which have no impact on weak trace equivalence, are added by the transformation. □

### 3.2 Connecting Choices

*Unique points of choice* ensure that in a choice all the participants agree on the chosen branch. Intuitively, unique points of choice can be guaranteed by ensuring that a single participant performs the choice and informs all the other involved participants.

**Definition 3** (Unique points of choice). *A choreography $C$ has* unique points of choice *iff each subterm of the form $C' + C''$ satisfies:*

1. $\forall \mathtt{a} \rightarrow \mathtt{b}\!:\!o_1^? \in \text{transI}(C'), \forall \mathtt{c} \rightarrow \mathtt{d}\!:\!o_2^? \in \text{transI}(C'').\mathtt{a} = \mathtt{c}$*;*
2. $\text{roles}(C') = \text{roles}(C'')$*;*

*where* $\text{roles}(\bullet)$ *computes the set of roles in a choreography.*

We present below a pattern able to ensure unique points of choice. We apply the pattern to all the subterms of the form $C' + C''$ that do not satisfy one of the conditions. Take one such term $C' + C''$. If condition 1 is not satisfied then choose a fresh role $\mathtt{e}$. Consider all the interactions $\mathtt{a} \to \mathtt{b}\!:\!o^?$ contributing to $\mathrm{transI}(C')$ or to $\mathrm{transI}(C'')$. For each of them choose a fresh operation $o^*_f$ and replace $\mathtt{a} \to \mathtt{b}\!:\!o^?$ with $\mathtt{e} \to \mathtt{a}\!:\!o^*_f;\ \mathtt{a} \to \mathtt{b}\!:\!o^?$.

Suppose now that condition 1 is satisfied, while condition 2 is not. Then we can assume a role $\mathtt{e}$ which is the sender of all the interactions in $\mathrm{transI}(C' + C'')$. Consider each role $\mathtt{a}$ that occurs in $C'$ but not in $C''$ (the symmetric case is analogous). For each of them add in parallel to $C''$ the interaction $\mathtt{e} \to \mathtt{a}\!:\!o^*_f$ where $o^*_f$ is a fresh operation.

**Proposition 2.** *Given a choreography $C$ we can derive using the pattern above a choreography $C'$ which has unique points of choice such that $C$ and $C'$ are weak trace equivalent.*

*Proof.* We have to apply the pattern to all the subterms failing to satisfy the condition. We start from smaller subterms, and then move to larger ones. We have to show that at the end all the subterms satisfy the conditions. We consider the transformation ensuring condition 1 first, then the one ensuring condition 2.

For the first transformation, given a subterm $D' + D''$ there are two cases: either some initial interactions inside $D'$ and $D''$ have been modified or not. In the second case the thesis follows by inductive hypothesis. In the first case all the initial interactions have been changed, and the freshly introduced role is the new sender in all of them. Thus the condition is satisfied.

Let us consider the second transformation. The transformation has no impact on subterms, since it only adds interactions in parallel. For the whole term the thesis holds by construction. Note that the term continues to satisfy the other condition.

Weak trace equivalence is ensured since only interactions on private operations, which have no impact on weak trace equivalence, are added by the transformation. □

## 3.3 Connecting Repeated Operations

If different interactions use the same operation, one has to ensure that messages do not get mixed, i.e. that the output of one interaction is not matched with the input of a different interaction on the same operation. Since in our model outputs do not contain the target participant, and inputs do not contain the expected sender, the problem is particularly relevant. The same problem however may occur also if the output contains the target participant and the input the expected sender, but only for interactions between the same participants, as shown by the example below.

**Example 1.** *Consider the choreography $C$ below:*

$$C = (\mathtt{a} \to \mathtt{b}\!:\!o_1; \mathtt{b} \to \mathtt{c}\!:\!o; \mathtt{c} \to \mathtt{d}\!:\!o_2) \parallel (\mathtt{a} \to \mathtt{b}\!:\!o_3; \mathtt{b} \to \mathtt{c}\!:\!o; \mathtt{c} \to \mathtt{d}\!:\!o_4)$$

*The two interactions $\mathtt{b} \to \mathtt{c}\!:\!o$, both between role $\mathtt{b}$ and role $\mathtt{c}$, may interfere, since the send from one of them could be received by the other one.*

Clearly, given two interactions on the same operation, the problem does not occur if the output of the first is never enabled together with the input of the second, and vice versa.

To ensure this, in [7] we required causal dependencies between the input in one interaction and the outputs of different interactions on the same operation, and vice versa. To formalize this concept we introduce the notion of *event*. For each interaction $\mathtt{a} \to \mathtt{b}\!:\!o^?$ we distinguish two events: a sending event at role $\mathtt{a}$ and a receiving event at role $\mathtt{b}$. The sending event and the receiving event in the same interaction are called matching events. We use $e$ to range over events, $s$ to range over sending events and $r$ to range over receiving events. We denote by $\overline{e}$ the event matching event $e$. If event $e$ precedes event $e'$ in the causal relation then $e'$ can become enabled only after $e$ has been executed, thus they cannot be matched. However, this requirement alone is not suitable for our aims, since this condition is too strong to be enforced by a transformation preserving weak traces.

Luckily, also events in a suitable relation of conflict cannot be matched. Requiring that events are in opposite branches of a choice, however, is not enough, since they may be both enabled. For instance, in $\mathtt{a} \to \mathtt{b}\!:\!o + \mathtt{c} \to \mathtt{d}\!:\!o$ the output from $\mathtt{a}$ can be captured by $\mathtt{d}$[1]. However, if the role containing the event is already aware of the choice, then the events are never enabled together and thus cannot be matched, since when one of them becomes enabled the other one has already been discarded. We call *full conflict* such a relation. For instance, in

$$(\mathtt{a} \to \mathtt{b}\!:\!o';\mathtt{b} \to \mathtt{a}\!:\!o;\mathtt{a} \to \mathtt{c}\!:\!o') + (\mathtt{a} \to \mathtt{b}\!:\!o'';\mathtt{b} \to \mathtt{a}\!:\!o;\mathtt{a} \to \mathtt{c}\!:\!o'')$$

the two interactions on $o$ do not interfere. Note that an interference could cause the wrong continuation to be chosen.

Thus we require here that a send and a receive in different interactions but on the same operation are either in a causality relation or in a full conflict relation. We call this condition *causality safety*. Causality safety can be enforced by a transformation preserving weak traces. We refer to [8] for the proof that causality safety ensures the desired properties of choreography projection.

To define causality safety we thus need to define a *causality relation* and a *full conflict relation*.

**Definition 4** (Causality relation)**.** *Let us consider a choreography $C$. A* causality relation $\leq_C$ *is a partial order among events of $C$. We define $\leq_C$ as the minimum partial order satisfying:*

**sequentiality:** *for each subterm of the form $C';C''$ and each role* $\mathtt{a}$*, if $r'$ is a receive event in $C'$ at role* $\mathtt{a}$*, $e''$ is a generic event in $C''$ at role* $\mathtt{a}$ *then $r' \leq_C e''$;*

**synchronization:** *for each receive event $r$ and generic event $e'$, $r \leq_C e'$ implies $\overline{r} \leq_C e'$.*

**Definition 5** (Full conflict relation)**.** *Let us consider a choreography $C$. A* full conflict relation ${}^{\mathrm{f}}\#_C$ *is a relation among events of $C$. We define ${}^{\mathrm{f}}\#_C$ as the smallest symmetric relation satisfying:*

**choice:** *for each subterm of the form $C' + C''$ and each role* $\mathtt{a}$*, if $e'$ is an event in $C'$ at role* $\mathtt{a}$*, $e''$ is an event in $C''$ at role* $\mathtt{a}$ *then $e'\, {}^{\mathrm{f}}\#_C\, e''$;*

**causality:** *if $e'\, {}^{\mathrm{f}}\#_C\, e''$ and $e'' \leq_C e'''$ then $e'\, {}^{\mathrm{f}}\#_C\, e'''$.*

We can now define causality safety.

**Definition 6** (Causality safety)**.** *A choreography $C$ is* causality safe *iff for each pair of interactions $\mathtt{a} \to \mathtt{b}\!:\!o^?$ (with events $s_1$ and $r_1$) and $\mathtt{c} \to \mathtt{d}\!:\!o^?$ (with events $s_2$ and $r_2$) on the same operation $o^?$ we have that:*

- $s_1 \leq_C r_2 \vee r_2 \leq_C s_1 \vee s_1\, {}^{\mathrm{f}}\#_C\, r_2$*;*
- $s_2 \leq_C r_1 \vee r_1 \leq_C s_2 \vee s_2\, {}^{\mathrm{f}}\#_C\, r_1$.

As an example, choreography $C = \mathtt{a} \to \mathtt{b}\!:\!o \parallel \mathtt{c} \to \mathtt{d}\!:\!o$ is not causality safe, since the output on $o$ at $\mathtt{a}$ is enabled together with the input on $o$ at $\mathtt{d}$, thus enabling a communication from $\mathtt{a}$ to $\mathtt{d}$ which should not be allowed.

[1] This choreography however has not unique points of choice.

A causality safety issue is immediately solved by renaming one of the operations. However, this changes the specification. We show how to solve the causality safety issue while sticking to the original (weak) behavior. We have different approaches according to the top-level operator of the smaller term including the conflicting interactions. Thus we distinguish *parallel causality safety*, *sequential causality safety* and *choice causality safety*.

To solve parallel causality safety issues we apply a form of *expansion law* that transforms the parallel composition into nondeterminism, thus either removing completely the causality safety issue or transforming it into sequential or choice causality safety, discussed later on. Using the expansion law one can transform any choreography into a *normal form* defined as below.

**Definition 7** (Normal form)**.** *A choreography C is in* normal form *if it is written as:*

$$\sum_i \mathtt{a}_i \rightarrow \mathtt{b}_i : o_i^?;C_i$$

*where $\sum_i$ is n-ary nondeterministic choice (we can see the empty sum as* **0***), and $C_i$ is in normal form for each i.*

The expansion law is defined below.

**Definition 8** (Expansion law)**.**

$$\begin{aligned}(\sum_i \mathtt{a}_i \rightarrow \mathtt{b}_i : o_i^?;\, C_i) \parallel (\sum_j \mathtt{a}_j \rightarrow \mathtt{b}_j : o_j^?;C_j) \quad = \quad & (\sum_i \mathtt{a}_i \rightarrow \mathtt{b}_i : o_i^?;(C_i \parallel (\sum_j \mathtt{a}_j \rightarrow \mathtt{b}_j : o_j^?;C_j))) \\ & +(\sum_j \mathtt{a}_j \rightarrow \mathtt{b}_j : o_j^?;(C_j \parallel (\sum_i \mathtt{a}_i \rightarrow \mathtt{b}_i : o_i^?;C_i)))\end{aligned}$$

The expansion law is correct w.r.t. trace equivalence, in the sense that applying the expansion law does not change the set of traces (neither strong nor weak).

**Lemma 1.** *Let $C$ and $C'$ be choreographies, with $C'$ obtained by applying the expansion law to a subterm of $C$. Then $C$ and $C'$ are (strong and weak) trace equivalent.*

*Proof.* Labels not involving the subterm are easily mimicked. Consider the first label involving the subterm. If no such label exists then the thesis follows. Otherwise, the label corresponds to the execution of one of the interactions $\mathtt{a}_i \rightarrow \mathtt{b}_i : o_i^?$ or $\mathtt{a}_j \rightarrow \mathtt{b}_j : o_j^?$. Executing any of these interactions reduces both the terms to the same term. The thesis follows. □

Using the expansion law we can put any choreography $C$ in normal form.

**Proposition 3** (Normalization)**.** *Given a choreography $C$ there is a choreography $C'$ in normal form such that $C$ and $C'$ are weak trace equivalent.*

*Proof.* The proof is by structural induction on the number of interactions occurring in $C$. The cases of interactions and **0** are trivial. Choreography **1** can be replaced by any interaction on a private operation without changing the set of weak traces. For sequential composition note that $(\sum_i \mathtt{a}_i \rightarrow \mathtt{b}_i : o_i^?;C_i);C'$ and $(\sum_i \mathtt{a}_i \rightarrow \mathtt{b}_i : o_i^?;C_i;C')$ have the same set of traces. $C_i;C'$ can be transformed in normal form by inductive hypothesis. For nondeterministic choice the thesis is trivial (it is easy to check that nondeterministic choice is associative). For parallel composition one can apply the expansion law, and the thesis follows from Lemma 1 and inductive hypothesis. □

Let us now consider sequential causality safety. The term should have the form $C';C''$, with an interaction $\mathtt{a} \to \mathtt{b}:o^?$ in $C'$ and an interaction $\mathtt{c} \to \mathtt{d}:o^?$ in $C''$. If the term satisfies connectedness for sequence then the send at $\mathtt{c}$ cannot become enabled before the receive at $\mathtt{b}$. We thus have to ensure that the receive at $\mathtt{d}$ becomes enabled after the message from the send at $\mathtt{a}$ has been received. Choose fresh operations $o_f^*$ and $o_g^*$ and replace $\mathtt{a} \to \mathtt{b}:o^?$ with $\mathtt{a} \to \mathtt{b}:o^?;\mathtt{b} \to \mathtt{d}:o_f^*;\mathtt{d} \to \mathtt{b}:o_g^*$.

The pattern for choice causality safety is similar. The term should have the form $C'+C''$, with an interaction $\mathtt{a} \to \mathtt{b}:o^?$ in $C'$ and an interaction $\mathtt{c} \to \mathtt{d}:o^?$ in $C''$. Assume that there is an interference between the send at $\mathtt{a}$ and the receive at $\mathtt{d}$ (the symmetric case is similar). Choose fresh operations $o_f^*$ and $o_g^*$ and replace $\mathtt{c} \to \mathtt{d}:o^?$ with $\mathtt{c} \to \mathtt{d}:o_f^*;\mathtt{d} \to \mathtt{c}:o_g^*;\mathtt{c} \to \mathtt{d}:o^?$

To prove the correctness of the transformation we need two auxiliary lemmas.

**Lemma 2.** *Let $C$ be a choreography which is connected for sequence. Then all sending events in $C$ depend on sending events in initial interactions in $C$.*

*Proof.* By structural induction on $C$. The only difficult case is sequential composition, when $C = C';C''$. For events in $C'$ the thesis follows by inductive hypothesis. For events in $C''$, if they are in an initial interaction, from connectedness for sequence they are in the same role as receiving events in the final interactions in $C'$, thus they depend on them. By synchronization they also depend on sending events in $C'$. The thesis follows by induction and by transitivity. For events not in initial interactions in $C''$, by inductive hypothesis they depend on events in initial interactions in $C''$, thus the thesis follows from what proved above by transitivity. □

**Lemma 3.** *Let $C = C';C''$ be a choreography which is connected for sequence. Then there are no causality safety issues between receiving events in $C'$ and sending events in $C''$.*

*Proof.* From connectedness for sequence all the receiving events in final interactions of $C'$ and all the sending events in initial interactions of $C''$ are performed by the same role. Thus, from the sequentiality condition in the definition of causality relation, they are causally related. Now the more difficult case is when the send $s$ in $C''$ is not initial and the receive $r$ in $C'$ is not final. Thanks to Lemma 2 the send $s$ depends on an initial send, and by transitivity on each final receive. Thanks to synchronization it depends also on each final send. Assume $r$ is not final. Thanks to connectedness for sequence it is in the same role of a following send $s'$, thus $s'$ depends on $r$. If $s'$ is final the thesis follows. Otherwise we iterate the procedure on a smaller term, and the thesis follows by induction. □

We can now prove the correctness of the transformation.

**Proposition 4.** *Let $C$ be a choreography which is connected for sequence, has unique points of choice and is parallel causality safe. We can derive using the pattern above a choreography $C'$ which has no sequential or choice causality safety issue such that $C$ and $C'$ are weak trace equivalent. Also, $C'$ is still connected for sequence, has unique points of choice and is parallel causality safe.*

*Proof.* We prove the thesis for just one causality safety issue, the more general result follows by iteration. Take a causality safety issue, and consider the smallest subterm including the two conflicting interactions. We have two cases according to whether the subterm has the form $C';C''$ or $C'+C''$.

Let us consider the first case. Thanks to Lemma 3 the issue is between the send at $\mathtt{a}$ in an interaction $\mathtt{a} \to \mathtt{b}:o^?$ in $C'$ and a receive at $\mathtt{d}$ in an interaction $\mathtt{c} \to \mathtt{d}:o^?$ in $C''$.

After the transformation, the receive at $\mathtt{d}$ in $\mathtt{c} \to \mathtt{d}:o^?$ depends on the receive at $\mathtt{d}$ in $\mathtt{b} \to \mathtt{d}:o_f^*$ by sequentiality. The receive at $\mathtt{d}$ in $\mathtt{b} \to \mathtt{d}:o_f^*$ depends on the send at $\mathtt{b}$ inside the same interaction by synchronization, on the receive at $\mathtt{b}$ in $\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?$ by sequentiality, and on the send at $\mathtt{a}$ inside the same interaction again by synchronization. This proves that the issue is solved.

Let us consider the second case. Assume that the issue is between a send in an interaction $\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?$ in $C'$ and a receive in an interaction $\mathtt{c} \rightarrow \mathtt{d}\!:\!o^?$ in $C''$. The sends at $\mathtt{a}$ and at $\mathtt{c}$ are causally dependent on sends in an initial interaction of the term thanks to Lemma 2. The transformation adds a dependency between the send at $\mathtt{c}$ and the receive at $\mathtt{d}$ inside the same interaction. Thus the send at $\mathtt{a}$ and the receive at $\mathtt{d}$ are in full conflict, since they are both causally dependent on sends from initial interactions of the term, which are all in the same role thanks to unique points of choice. Again, this proves that the issue is solved.

Weak trace equivalence is ensured since only interactions on private operations, which have no impact on weak trace equivalence, are added by the transformation.

We have to show that the other properties are preserved. All of them hold by construction for the newly introduced terms. Connectedness for sequence is preserved since the transformation preserves senders of initial interactions and receivers of final interactions. Unique points of choice are preserved for the same reason, and since the transformation does not add new roles. Parallel causality safety is preserved since the transformation only introduces interactions on fresh operations. □

## 3.4 Combining the Amending Techniques

Till now we have shown that given a choreography which fails to satisfy one of the connectedness conditions, we can transform it into an equivalent one that satisfies the chosen connectedness condition. Some care is required to avoid that, while ensuring that one condition is satisfied, violations of other conditions are introduced. In fact, if such a situation would occur repeatedly, the connecting procedure may not terminate.

The following theorem proves that we can combine the connecting patterns presented in the previous sections to define a terminating algorithm transforming any choreography into a connected choreography.

**Theorem 1** (Connecting choreographies)**.** *There is a terminating procedure that given any choreography $C$ creates a new choreography $D$ such that:*

- *$D$ is connected;*
- *$C$ and $D$ are weak trace equivalent.*

*Proof.* We can apply the normalization procedure to all the subterms of choreography $C$ that do not satisfy parallel causality safety, starting from the smallest subterms to the largest, to get a choreography $C'$ which is parallel causality safe (since the undesired parallel compositions have been removed) and which is weak trace equivalent to $C$ thanks to Proposition 3.

Now, again from the smallest subterms to the largest, we can apply to $C'$ the procedure for ensuring unique points of choice to those subterms which have a top-level nondeterministic choice operator, and the procedure for making them connected for sequence to those subterms which have a top-level sequential composition operator obtaining a choreography $C''$.

For terms of the first kind, thanks to Proposition 2, we obtain terms which have unique points of choice. They are also parallel causality safe and connected for sequence, since the transformation may not create these issues. The same holds for terms of the second kind thanks to Proposition 1. In both the cases, the resulting term is weak trace equivalent to the starting one. By transitivity $C''$ is weak trace equivalent to $C$.

Finally, to each sequential and choice causality safety issue we can apply the procedure to solve it. From Proposition 4 we know that the resulting choreography $D$ has no sequential or choice causality safety issue, still satisfies the other properties and is weak trace equivalent to $C''$. The thesis follows by transitivity. □

**Example 2.** *We now apply our procedure to the paradigmatic choreography $C = \mathtt{a} \rightarrow \mathtt{b}\!:\!o \parallel \mathtt{c} \rightarrow \mathtt{d}\!:\!o$. First note that $C$ does not satisfy parallel causality safety. By application of the expansion law we obtain:*

$$C_1 = \mathtt{a} \rightarrow \mathtt{b}\!:\!o; \mathtt{c} \rightarrow \mathtt{d}\!:\!o + \mathtt{c} \rightarrow \mathtt{d}\!:\!o; \mathtt{a} \rightarrow \mathtt{b}\!:\!o$$

*Proceeding from smaller to larger subterms, we first encounter the subterms $\mathtt{a} \rightarrow \mathtt{b}\!:\!o; \mathtt{c} \rightarrow \mathtt{d}\!:\!o$ and $\mathtt{c} \rightarrow \mathtt{d}\!:\!o; \mathtt{a} \rightarrow \mathtt{b}\!:\!o$ which are not connected for sequence (and are not sequential causality safe). By applying the corresponding pattern to the two subterms, we obtain:*

$$C_2 = \mathtt{a} \rightarrow \mathtt{b}\!:\!o; \mathtt{b} \rightarrow \mathtt{e}'\!:\!o_1^*; \mathtt{e}' \rightarrow \mathtt{c}\!:\!o_2^*; \mathtt{c} \rightarrow \mathtt{d}\!:\!o + \mathtt{c} \rightarrow \mathtt{d}\!:\!o; \mathtt{d} \rightarrow \mathtt{e}''\!:\!o_3^*; \mathtt{e}'' \rightarrow \mathtt{a}\!:\!o_4^*; \mathtt{a} \rightarrow \mathtt{b}\!:\!o$$

*Now, the whole term does not have unique points of choice, and is not choice causality safe. By applying the first of the transformations ensuring unique points of choice, we obtain:*

$$\begin{aligned} C_3 \quad = \quad & \mathtt{e} \rightarrow \mathtt{a}\!:\!o_5^*; \mathtt{a} \rightarrow \mathtt{b}\!:\!o; \mathtt{b} \rightarrow \mathtt{e}'\!:\!o_1^*; \mathtt{e}' \rightarrow \mathtt{c}\!:\!o_2^*; \mathtt{c} \rightarrow \mathtt{d}\!:\!o + \\ & \mathtt{e} \rightarrow \mathtt{c}\!:\!o_6^*; \mathtt{c} \rightarrow \mathtt{d}\!:\!o; \mathtt{d} \rightarrow \mathtt{e}''\!:\!o_3^*; \mathtt{e}'' \rightarrow \mathtt{a}\!:\!o_4^*; \mathtt{a} \rightarrow \mathtt{b}\!:\!o \end{aligned}$$

*By applying the transformation ensuring that both the branches have the same set of roles, we obtain:*

$$\begin{aligned} C_4 \quad = \quad & \big(\mathtt{e} \rightarrow \mathtt{a}\!:\!o_5^*; \mathtt{a} \rightarrow \mathtt{b}\!:\!o; \mathtt{b} \rightarrow \mathtt{e}'\!:\!o_1^*; \mathtt{e}' \rightarrow \mathtt{c}\!:\!o_2^*; \mathtt{c} \rightarrow \mathtt{d}\!:\!o \parallel \mathtt{e} \rightarrow \mathtt{e}''\!:\!o_7^*\big) + \\ & \big(\mathtt{e} \rightarrow \mathtt{c}\!:\!o_6^*; \mathtt{c} \rightarrow \mathtt{d}\!:\!o; \mathtt{d} \rightarrow \mathtt{e}''\!:\!o_3^*; \mathtt{e}'' \rightarrow \mathtt{a}\!:\!o_4^*; \mathtt{a} \rightarrow \mathtt{b}\!:\!o \parallel \mathtt{e} \rightarrow \mathtt{e}'\!:\!o_8^*\big) \end{aligned}$$

*We are now left with sequential and choice causality safety issues. We start by solving the sequential ones, which are one for each branch. As expected, they are between the sender of the first interaction on $o$ and the receiver of the second. By applying the pattern we get:*

$$\begin{aligned} C_5 \quad = \quad & \big(\mathtt{e} \rightarrow \mathtt{a}\!:\!o_5^*; \mathtt{a} \rightarrow \mathtt{b}\!:\!o; \mathtt{b} \rightarrow \mathtt{d}\!:\!o_9^*; \mathtt{d} \rightarrow \mathtt{b}\!:\!o_{10}^*; \mathtt{b} \rightarrow \mathtt{e}'\!:\!o_1^*; \mathtt{e}' \rightarrow \mathtt{c}\!:\!o_2^*; \mathtt{c} \rightarrow \mathtt{d}\!:\!o \parallel \mathtt{e} \rightarrow \mathtt{e}''\!:\!o_7^*\big) + \\ & \big(\mathtt{e} \rightarrow \mathtt{c}\!:\!o_6^*; \mathtt{c} \rightarrow \mathtt{d}\!:\!o; \mathtt{d} \rightarrow \mathtt{b}\!:\!o_{11}^*; \mathtt{b} \rightarrow \mathtt{d}\!:\!o_{12}^*; \mathtt{d} \rightarrow \mathtt{e}''\!:\!o_3^*; \mathtt{e}'' \rightarrow \mathtt{a}\!:\!o_4^*; \mathtt{a} \rightarrow \mathtt{b}\!:\!o \parallel \mathtt{e} \rightarrow \mathtt{e}'\!:\!o_8^*\big) \end{aligned}$$

*We are left with choice causality safety issues between the interaction $\mathtt{a} \rightarrow \mathtt{b}\!:\!o$ in the first branch and the interaction $\mathtt{c} \rightarrow \mathtt{d}\!:\!o$ in the second branch. By applying the pattern we get:*

$$\begin{aligned} C_6 \quad = \quad & \big(\mathtt{e} \rightarrow \mathtt{a}\!:\!o_5^*; \mathtt{a} \rightarrow \mathtt{b}\!:\!o_{13}^*; \mathtt{b} \rightarrow \mathtt{a}\!:\!o_{14}^*; \mathtt{a} \rightarrow \mathtt{b}\!:\!o; \mathtt{b} \rightarrow \mathtt{d}\!:\!o_9^*; \mathtt{d} \rightarrow \mathtt{b}\!:\!o_{10}^*; \\ & \quad \mathtt{b} \rightarrow \mathtt{e}'\!:\!o_1^*; \mathtt{e}' \rightarrow \mathtt{c}\!:\!o_2^*; \mathtt{c} \rightarrow \mathtt{d}\!:\!o \parallel \mathtt{e} \rightarrow \mathtt{e}''\!:\!o_7^*\big) + \\ & \big(\mathtt{e} \rightarrow \mathtt{c}\!:\!o_6^*; \mathtt{c} \rightarrow \mathtt{d}\!:\!o_{15}^*; \mathtt{d} \rightarrow \mathtt{c}\!:\!o_{16}^*; \mathtt{c} \rightarrow \mathtt{d}\!:\!o; \mathtt{d} \rightarrow \mathtt{b}\!:\!o_{11}^*; \mathtt{b} \rightarrow \mathtt{d}\!:\!o_{12}^*; \\ & \quad \mathtt{d} \rightarrow \mathtt{e}''\!:\!o_3^*; \mathtt{e}'' \rightarrow \mathtt{a}\!:\!o_4^*; \mathtt{a} \rightarrow \mathtt{b}\!:\!o \parallel \mathtt{e} \rightarrow \mathtt{e}'\!:\!o_8^*\big) \end{aligned}$$

*This example shows that solving a parallel causality safety issue may require a considerable amount of auxiliary communications. Thus, it is advised to change the name of the operation if possible. If not possible, our approach will solve the problem. Other issues are solved more easily, since they involve no duplication of terms.*

## 4 Application: Two-Buyers Protocol

We show now how our transformation for connecting choreographies can be used as an effective design tool for programming multiparty choreographies. We model the example reported in [6], the two-buyers protocol, where two buyers – $\mathtt{b}_1$ and $\mathtt{b}_2$ – combine their finances for buying a product from a seller $\mathtt{s}$. The protocol starts with $\mathtt{b}_1$ asking the price for the product of interest to $\mathtt{s}$. Then, $\mathtt{s}$ communicates the price to both $\mathtt{b}_1$ and $\mathtt{b}_2$. Subsequently, $\mathtt{b}_1$ notifies $\mathtt{b}_2$ of how much she is willing to contribute to the purchase. Finally, $\mathtt{b}_2$ chooses whether to confirm the purchase and ask $\mathtt{s}$ to send a delivery date for the product; otherwise, the choreography terminates. We do not deal here with how this choice is performed, as our choreographies abstract from data.

To create a quick prototype choreography $C$ for the two-buyers protocol, we focus only on the main interactions. When writing the choreography we do not worry about connectedness conditions, since we will enforce them automatically later on. The code follows naturally:

$$\begin{aligned} C = \quad & \mathtt{b}_1 \rightarrow \mathtt{s} : \mathsf{price};\ (\ \mathtt{s} \rightarrow \mathtt{b}_1 : \mathsf{quote1}\ \parallel\ \mathtt{s} \rightarrow \mathtt{b}_2 : \mathsf{quote2}\ );\ \mathtt{b}_1 \rightarrow \mathtt{b}_2 : \mathsf{contrib}; \\ & (\ \mathtt{b}_2 \rightarrow \mathtt{s} : \mathsf{ok}; \mathtt{s} \rightarrow \mathtt{b}_2 : \mathsf{delivery}\ +\ \mathbf{1}\ ) \end{aligned}$$

The code above is just a direct translation of our explanation in natural language into a choreography. We can immediately observe that the choreography is not connected, since it contains 2 violations of the connectedness conditions:

- the subterm $(\ \mathtt{s} \rightarrow \mathtt{b}_1 : \mathsf{quote1}\ \parallel\ \mathtt{s} \rightarrow \mathtt{b}_2 : \mathsf{quote2}\ );\ \mathtt{b}_1 \rightarrow \mathtt{b}_2 : \mathsf{contrib}$ is not connected for sequence; thus, e.g., $\mathtt{b}_1$ may send the $\mathsf{contrib}$ message before $\mathtt{b}_2$ receives the message for $\mathsf{quote2}$[2];
- the subterm $(\ \mathtt{b}_2 \rightarrow \mathtt{s} : \mathsf{ok}; \mathtt{s} \rightarrow \mathtt{b}_2 : \mathsf{delivery}\ +\ \mathbf{1}\ )$ has not unique points of choice.

We can apply our transformation to amend our choreography prototype, transforming it into a connected choreography which is weak trace equivalent to $C$, obtaining:

$$\begin{aligned} C_1 = \quad & \mathtt{b}_1 \rightarrow \mathtt{s} : \mathsf{price};\ (\ \mathtt{s} \rightarrow \mathtt{b}_1 : \mathsf{quote1};\ \mathtt{b}_1 \rightarrow \mathtt{e}_1 : o_1^*\ \parallel\ \mathtt{s} \rightarrow \mathtt{b}_2 : \mathsf{quote2};\ \mathtt{b}_2 \rightarrow \mathtt{e}_1 : o_2^*\ ); \\ & \mathtt{e}_1 \rightarrow \mathtt{b}_1 : o_3^*;\ \mathtt{b}_1 \rightarrow \mathtt{b}_2 : \mathsf{contrib};\ (\ \mathtt{b}_2 \rightarrow \mathtt{s} : \mathsf{ok}; \mathtt{s} \rightarrow \mathtt{b}_2 : \mathsf{delivery}\ + (\ \mathbf{1} \parallel \mathtt{b}_2 \rightarrow \mathtt{s} : o_4^*\ )\ ) \end{aligned}$$

The choreography $C_1$ above is connected, thus it can be projected, and the projection will be trace equivalent to $C_1$ itself (thanks to the results in [7]), and weak trace equivalent to the original choreography $C$.

## 5 Conclusions

In previous work [7] we have defined syntactic conditions that guarantee choreography connectedness, i.e. that the endpoints obtained by the natural projection of a choreography correctly implement the global specification given by the choreography itself. In this paper we have presented a procedure for amending non-connected choreographies by reducing parallelism and adding interactions on private operations in such a way that all the above conditions are satisfied, while preserving the observational semantics. To the best of our knowledge, only two other papers consider the possibility to add messages to a choreography in order to make it correctly projectable [10, 9].

In [10] non-connected sequences are connected by adding a synchronization from every role involved in a final interaction before the sequential composition to every role involved in an initial interaction after the sequential composition, while non-connected choices are modified by selecting a dominant role responsible for taking the choice and communicating it to the other participants. Our approach reduces the number of added synchronizations in the case of sequential compositions, and allows for a symmetric projection that treats all the roles in the same way in the case of choices. In [10] there is no discussion

[2] This may happen only with the asynchronous semantics.

about repeated operations, and to the best of our understanding of the paper this is problematic. In fact, the choreography $C$ in Example 1 (written according to our syntax, which is slightly different w.r.t. the one adopted in [10]) is well-formed according to the conditions in [10], but it is not projectable.

*Collaboration* and *Sequence diagrams* are popular visual representations of choreographies [5]. In this area, the work more similar to ours is [9], where the problem of amending collaboration diagrams is addressed. In collaboration diagrams the interactions are labels of edges in a graph which has one node for each role, and one edge for each pair of interacting roles. Interactions are organized in threads representing sub-choreographies. Messages are totally ordered within the same thread, while a partial order can be defined among interactions belonging to different threads. The problem of amending non-connected choreographies is simplified in this context: collaboration diagrams do not have a global choice composition operator among sub-choreographies, and two distinct threads use two disjoint sets of operations. For this reason, only the problem of non-connected sequences is addressed in [9].

Another paper amending choreographies is [2], however they consider a different problem. In fact, they consider choreographies including annotations on the communicated data, and they concentrate on amending those annotations.

Possible extensions of the present work include its application to existing choreography languages, such as [4], where however the intended semantics of sequential composition is weaker, since independent interactions can be freely swapped. The main difficulty in extending the approach is dealing with infinite behaviors. Having an iteration construct should not be a problem: roughly one has to make connected the choice between iterating and exiting the iteration using the techniques for choice, and the sequential execution of different iterations using the techniques for sequence. General recursion would be more difficult to deal with, since subterms belonging to different iterations may execute in parallel. Adding data to our language would not change the issues discussed in the present paper.

## References

[1] Laura Bocchi, Kohei Honda, Emilio Tuosto & Nobuko Yoshida (2010): *A Theory of Design-by-Contract for Distributed Multiparty Interactions*. In: *Proc. of CONCUR 2010, LNCS* 6269, Springer, pp. 162–176, doi:10.1007/978-3-642-15375-4_12.

[2] Laura Bocchi, Julien Lange & Emilio Tuosto (2012): *Three Algorithms and a Methodology for Amending Contracts for Choreographies*. *Sci. Ann. Comp. Sci.* 22(1), pp. 61–104, doi:10.7561/SACS.2012.1.61.

[3] Marco Carbone, Kohei Honda & Nobuko Yoshida (2007): *Structured Communication-Centred Programming for Web Services*. In: *Proc. of ESOP'07*, *LNCS* 4421, Springer, pp. 2–17, doi:10.1007/978-3-540-71316-6_2.

[4] Marco Carbone & Fabrizio Montesi (2013): *Deadlock-freedom-by-design: multiparty asynchronous global programming*. In: *Proc. of POPL 2013*, ACM, pp. 263–274, doi:10.1145/2429069.2429101.

[5] Martin Fowler (2003): *UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition)*. Addison Wesley.

[6] Kohei Honda, Nobuko Yoshida & Marco Carbone (2008): *Multiparty asynchronous session types*. In: *Proc. of POPL'08*, ACM Press, pp. 273–284, doi:10.1145/1328438.1328472.

[7] Ivan Lanese, Claudio Guidi, Fabrizio Montesi & Gianluigi Zavattaro (2008): *Bridging the Gap between Interaction- and Process-Oriented Choreographies*. In: *Proc. of SEFM'08*, IEEE Press, pp. 323–332, doi:10.1109/SEFM.2008.11.

[8] Ivan Lanese, Fabrizio Montesi & Gianluigi Zavattaro: *Classifying Relationships between Interaction- and Process-Oriented Choreographies*. Available at http://www.cs.unibo.it/~lanese/publications/fulltext/ioc.pdf.

$$\textsc{(In)}\quad o^? \xrightarrow{o^?} \mathbf{1} \qquad \textsc{(Out)}\quad \overline{o^?} \xrightarrow{\overline{o^?}} \mathbf{1} \qquad \textsc{(Async-Out)}\quad \langle o^? \rangle \xrightarrow{\langle o^? \rangle} \mathbf{1} \qquad \textsc{(One)}\quad \mathbf{1} \xrightarrow{\surd} \mathbf{0}$$

$$\textsc{(Sequence)}\ \frac{P \xrightarrow{\gamma} P' \quad \gamma \neq \surd}{P;Q \xrightarrow{\gamma} P';Q} \qquad \textsc{(Inner Parallel)}\ \frac{P \xrightarrow{\gamma} P' \quad \gamma \neq \surd}{P \mid Q \xrightarrow{\gamma} P' \mid Q} \qquad \textsc{(Choice)}\ \frac{P \xrightarrow{\gamma} P'}{P + Q \xrightarrow{\gamma} P'} \qquad \textsc{(Seq-end)}\ \frac{P \xrightarrow{\surd} P' \quad Q \xrightarrow{\gamma} Q'}{P;Q \xrightarrow{\gamma} Q'}$$

$$\textsc{(Inner Par-end)}\ \frac{P \xrightarrow{\surd} P' \quad Q \xrightarrow{\surd} Q'}{P \mid Q \xrightarrow{\surd} P' \mid Q'} \qquad \textsc{(Lift)}\ \frac{P \xrightarrow{\gamma} P' \quad \gamma \neq \overline{o^?}, \surd}{[P]_{\mathtt{a}} \xrightarrow{\gamma:\mathtt{a}} [P']_{\mathtt{a}}} \qquad \textsc{(Lift-Tick)}\ \frac{P \xrightarrow{\surd} P'}{[P]_{\mathtt{a}} \xrightarrow{\surd} [P']_{\mathtt{a}}} \qquad \textsc{(Msg)}\ \frac{P \xrightarrow{\overline{o^?}} P'}{[P]_{\mathtt{a}} \xrightarrow{\overline{o^?}:\mathtt{a}} [P' \mid \langle o^? \rangle]_{\mathtt{a}}}$$

$$\textsc{(Synch)}\ \frac{S \xrightarrow{\langle o^? \rangle:\mathtt{a}} S' \quad S'' \xrightarrow{o^?:\mathtt{b}} S'''}{S \parallel S'' \xrightarrow{\mathtt{a} \to \mathtt{b}:o^?} S' \parallel S'''} \qquad \textsc{(Ext-Parallel)}\ \frac{S \xrightarrow{\gamma} S' \quad \gamma \neq \surd}{S \parallel S'' \xrightarrow{\gamma} S' \parallel S''} \qquad \textsc{(Ext-Par-End)}\ \frac{S \xrightarrow{\surd} S' \quad S'' \xrightarrow{\surd} S'''}{S \parallel S'' \xrightarrow{\surd} S' \parallel S'''}$$

Table 1: Projected systems asynchronous semantics (symmetric rules omitted).

[9] Gwen Salaun, Tevfik Bultan & Nima Roohi (2012): *Realizability of Choreographies Using Process Algebra Encodings*. *IEEE Transactions on Services Computing* 5(3), pp. 290–304, doi:10.1109/TSC.2011.9.

[10] Qiu Zongyan, Zhao Xiangpeng, Cai Chao & Yang Hongli (2007): *Towards the Theoretical Foundation of Choreography*. In: *Proc. of WWW'07*, ACM Press, pp. 973–982, doi:10.1145/1242572.1242704.

# A Projected Systems

We describe here the syntax and the operational semantics of *projected systems*. Projected systems, ranged over by $S$, $S'$, ..., are composed by *processes*, ranged over by $P$, $P'$, ..., describing the behavior of participants,

$$\begin{array}{rcl} P & ::= & o^? \mid \overline{o^?} \mid \mathbf{1} \mid P;P' \mid \ P \mid P' \ \mid P + P' \mid \langle o^? \rangle \mid \mathbf{0} \\ S & ::= & [P]_{\mathtt{a}} \mid S \parallel S' \end{array}$$

Processes include input action $o^?$ and output action $\overline{o^?}$ on a specific operation $o^?$ (either public or private), the empty process $\mathbf{1}$, sequential and parallel composition, and nondeterministic choice. The runtime syntax includes also messages $\langle o^? \rangle$, used in the definition of the asynchronous semantics, and the deadlocked process $\mathbf{0}$. Projected systems are parallel compositions of roles. Each role has a role name and executes a process. We require role names to be unique.

We define two LTS semantics for projected systems, one *synchronous* and one *asynchronous*. In the synchronous semantics input actions and output actions interact directly, while in the asynchronous one the sending event creates a message that, later, may interact with the corresponding input, generating a receiving event.

The asynchronous LTS for projected systems is the smallest LTS closed under the rules in Table 1. We use $\gamma$ to range over labels. Symmetric rules for parallel compositions (both internal and external) and choice have been omitted.

Rules IN and OUT execute input actions and output actions, respectively. Rule ASYNCH-OUT makes messages available for a corresponding input action. Rule ONE terminates an empty process. Rule SEQUENCE executes a step in the first component of a sequential composition. Rule INNER PARALLEL executes an action from a component of a parallel composition, while rule CHOICE starts the execution of an alternative in a nondeterministic choice. Rule SEQ-END acknowledges the termination of the first component of a sequential composition, starting the second component. Rule INNER PAR-END synchronizes the termination of two parallel components. Rule LIFT lifts actions to the system level, tagging them with the name of the role executing them. Action $\surd$ instead is dealt with by rule LIFT-TICK, which lifts it without adding the role name. Outputs instead are stored as messages by rule MSG. Rule SYNCH synchronizes a message with the corresponding input action, producing an interaction. Rule EXT-PARALLEL allows parallel systems to stay idle. Finally, rule EXT-PAR-END synchronizes the termination of parallel systems.

The synchronous LTS for projected systems is the smallest LTS closed under the rules in Table 1, where rules OUT, ASYNC-OUT and MSG are deleted and the new rule SYNC-OUT below is added:

$$\begin{array}{c}\text{(SYNC-OUT)}\\ \overline{o^?} \xrightarrow{\langle o^? \rangle}_s \mathbf{1}\end{array}$$

This rule allows outputs in the synchronous semantics to send messages that can directly interact with the corresponding input at the system level.

Synchronous transitions are denoted as $\xrightarrow{\gamma}_s$ instead of $\xrightarrow{\gamma}$, to distinguish them from the asynchronous ones.

As for choreographies, we define *traces*. We have different possibilities: in addition to the distinction between strong and weak traces, we distinguish *synchronous* and *asynchronous* traces.

**Definition 9** (Projected systems traces)**.** *A (strong maximal) synchronous trace of a projected system $S_1$ is a sequence of labels $\gamma_1,\dots,\gamma_n$, where $\gamma_i$ is of the form $\mathtt{a} \to \mathtt{b}\!:\!o^?$, or $\surd$ for each $i \in \{1,\dots,n\}$, such that there is a sequence of synchronous transitions $S_1 \xrightarrow{\gamma_1}_s \dots \xrightarrow{\gamma_n}_s S_{n+1}$ and such that $S_{n+1}$ has no outgoing transitions of the same form.*

*A (strong maximal) asynchronous trace of a projected system $S_1$ is a sequence of labels $\gamma_1,\dots,\gamma_n$, where $\gamma_i$ is of the form $\overline{o^?} : \mathtt{a}$, $\mathtt{a} \to \mathtt{b}\!:\!o^?$, or $\surd$ for each $i \in \{1,\dots,n\}$, such that there is a sequence of asynchronous transitions $S_1 \xrightarrow{\gamma_1} \dots \xrightarrow{\gamma_n} S_{n+1}$ and such that $S_{n+1}$ has no outgoing transitions of the same form. A weak (synchronous/asynchronous) trace of a projected system $S_1$ is obtained by removing all labels $\overline{o^*} : \mathtt{a}$ and $\mathtt{a} \to \mathtt{b}\!:\!o^*$ from a strong (synchronous/asynchronous) trace of $S_1$.*

In the definition of traces, input actions and messages are never considered, since they represent interactions with the external world, while we are interested in the behavior of closed systems.

We have shown in [7] that a connected choreography and the corresponding projected system have the same set of strong traces, if the synchronous semantics is used for the projected system. The correspondence with the asynchronous semantics is more complex, and we refer to [7] for a precise description.

# B Projection

In this section we show how to derive from a choreography $C$ a projected system $S$ implementing it. The idea is to project the choreography $C$ on the different roles, and build the system $S$ as parallel composition of the projections on the different roles. We consider here the most natural projection, which is essentially an homomorphism on most operators. As shown in [7], if $C$ satisfies the connectedness conditions, the resulting projected system is behaviorally related to the starting choreography $C$.

**Definition 10** (Projection function)**.** *Given a choreography $C$ and a role* $\mathtt{a}$*, the projection* $\mathrm{proj}(C,\mathtt{a})$ *of choreography $C$ on role* $\mathtt{a}$ *is defined by structural induction on $C$:*

$$\begin{array}{rcl}
\mathrm{proj}(\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?,\mathtt{a}) & = & \overline{o^?} \\
\mathrm{proj}(\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?,\mathtt{b}) & = & o^? \\
\mathrm{proj}(\mathtt{a} \rightarrow \mathtt{b}\!:\!o^?,\mathtt{c}) & = & \mathbf{1} \textit{ if } \mathtt{c} \neq \mathtt{a},\mathtt{b} \\
\mathrm{proj}(\mathbf{1},\mathtt{a}) & = & \mathbf{1} \\
\mathrm{proj}(\mathbf{0},\mathtt{a}) & = & \mathbf{0} \\
\mathrm{proj}(C;C',\mathtt{a}) & = & \mathrm{proj}(C,\mathtt{a});\mathrm{proj}(C',\mathtt{a}) \\
\mathrm{proj}(C \parallel C',\mathtt{a}) & = & \mathrm{proj}(C,\mathtt{a}) \mid \mathrm{proj}(C',\mathtt{a}) \\
\mathrm{proj}(C + C',\mathtt{a}) & = & \mathrm{proj}(C,\mathtt{a}) + \mathrm{proj}(C',\mathtt{a})
\end{array}$$

We denote with $\|_{i \in I} S_i$ the parallel composition of systems $S_i$ for each $i \in I$.

**Definition 11.** *Given a choreography $C$, the projection of $C$ is the system $S$ defined by:*

$$\mathrm{proj}(C) = \|_{\mathtt{a} \in \mathrm{roles}(C)} \left[\mathrm{proj}(C,\mathtt{a})\right]_{\mathtt{a}}$$

*where* $\mathrm{roles}(C)$ *is the set of roles in $C$.*