An abstract semantics of the global view of choreographies

Roberto Guanciale
KTH, Sweden
robertog@kth.se

Emilio Tuosto
University of Leicester, UK
emilio@le.ac.uk

We introduce an abstract semantics of the global view of choreographies. Our semantics is given in terms of pre-orders and can accommodate different lower level semantics. We discuss the adequacy of our model by considering its relation with communicating machines, that we use to formalise the local view. Interestingly, our framework seems to be more expressive than others where semantics of global views have been considered. This will be illustrated by discussing some interesting examples.

1 Introduction

The problem Choreographies have been advocated as a suitable methodology for the design and analysis of distributed applications. Roughly, a choreography describes how two of more distributed components coordinate with each other. Of course, in a distributed setting this coordination has to happen through exchange of messages. We embrace[1] the W3C's description [11]:

> Using the Web Services Choreography specification, a contract containing a global definition of the common ordering conditions and constraints under which messages are exchanged, is produced that describes, from a **global viewpoint** [...] observable behaviour [...]. Each party can then use the **global definition** to build and test solutions that conform to it. The global specification is in turn realised by combination of the resulting **local systems** [...]

This description conceptualise two views, a **global** and a **local** one, which enable the relations represented by the following diagram:

$$\text{Global view} \xrightarrow{\text{projection}} \text{Local view} \xleftarrow{\text{comply}} \text{Local systems} \tag{1}$$

where 'projection' is an operation producing the local view from the global one and 'comply' verifies that the behaviour of each components complies with the one of the corresponding local view.[2] For diagram (1) to make sense, precise semantics should be fixed for the global and the local views. The semantics of the latter is pretty understood: it directly emanates from the adopted communication model. In fact, the local view details how communications take place. For instance, in a channel-based communication model, the local view may specify what is the behaviour of each component in terms of its send/receive actions on channels.

What is instead "the" semantics of the global view? We investigate such question here. And, after making it more precise, we propose a new semantic framework for global views and discuss its advantages on existing frameworks.

[1] There are alternative interpretations of what a choreography is. See [2] for an interesting discussion and references.

[2] The 'projection' arrow in (1) may have an "inverse" one (cf. [12]), but this is immaterial here.

To appear in EPTCS.

A view of global views Although intriguing, the W3C description above, is not very enlightening to understand what a global view is; basically it says that a global view has to describe the observable behaviour from a global viewpoint...a bit too much circularity for a definition!

We will consider global views as high level descriptions of systems abstracting away some aspects in order to offer a *holistic* understanding of the communication behaviour of distributed systems. (We beg for the reader's patience: this is still vague, but will become precise in the forthcoming sections.) In a global view, components are not taken anymore in *isolation*. Rather they are *specified together*, while *forgetting of some details*. For us, this will mean to describe the protocol of interaction of a systems in a way that is oblivious of how messages are actually exchanged in the communication. For instance, in our example based on channels, the global view may abstract away from send/receive actions and use *interactions* as the unit of coordination [5].

The idea depicted in diagram (1) is beautiful. To the best of our knowledge, it has been firstly formally pursued in [10] and later followed by many others. The main reason that makes attractive the interplay in diagram (1) between global and local artefacts[3] is that it fosters some of the best principles of computer science:

Separation of concerns The *intrinsic logic* of the distributed coordination is expressed in and analysed on global artefacts, while the local artefacts refine such logic at lower levels of abstraction.

Modular software development lifecycle The W3C description above yields a distinctive element of choreographies which makes them appealing to practitioners. Choreographies allow independent development: components can harmoniously interact if they are proved to comply with the local view. Global and local views yield the "blueprints" of each component.

Principled design A choreographic framework orbits around the following implication:

if *cond*(global artefact) **then** *behave*(*projection*(global artefact))

that is, proving that a correctness condition *cond* holds on an abstraction (the global artefacts) guarantees that the system is well behaved, provided that the local artefacts are "compiled" from the global ones via a *projection* operation that preserves behaviour.

Therefore, providing good semantics for global artefacts is worthwhile: it gives precise algorithms and establishes precise relations between specifications of distributed systems (the global artefacts) and their refinements (the local artefacts).

Outline & Contributions We explain the advantages of defining an abstract semantics of global views in Section 2 and we give the syntax of our language of global artefacts in Section 3. Section 4 is the technical prelude; it introduces the notion of *reflection*, which is crucial for our generalisation. Section 5 yields another contribution: our abstract semantics of global artefacts. A first technical advantage of our semantics is provided by the definition of *well-branched*, explained through some the illustrative examples of Section 5. Our semantics is used in Section 6 to identify all licit traces of a choreography, thus making it possible to precisely identify the behaviors expected by the specification. Section 7 first recalls the communicating finite state machines (that are used to formalise the local behaviours) and then defines the projection of global artefacts on communicating machines. The main technical results establish that well-branched choreographies are deadlock free (Theorem 1) and that the executions specified by the global view contain those of its projections (Theorem 2) operation and shows that the local behaviours comply with the ones of the global specification. Concluding remarks are in Section 8.

[3] We will use the term 'artefact' when referring to actual specifications embodying the global/local views. Such embodiments may assume various forms: types [10], programs [8], graphs and automata [12, 9], executable models [11, 1], etc. Typically, the literature uses the (overloaded) word 'model' to refer to this flora of embodiments. We prefer the work 'artefact' because it allows us to refer to different contexts and different abstraction levels without attaching yet another meaning to 'model '.

2 Why going abstract?

As said, many authors have adopted the idea in diagram (1) and several semantics of (models of) global views have been introduced. We distinguish two broad classes.

Remark. *We mention a tiny portion of the literature in way of example; no claim of exhaustiveness.*

The largest class is possibly the one that includes the seminal work on global types [10]. The idea is that the semantics of global artefacts (incarnated by global types in [10]) is given in terms of the semantics of their local artefacts via a suitable projection operation. In the case of global types, the projection yields local types, that are process algebras equipped with an operational semantics. This approach to the semantics of global views is ubiquitous in the literature based on behavioural types, but it has also been adopted in [12] where global artefacts are global graphs [9] and local artefacts are communicating machines [4].

In the other class, the semantics of global views is defined explicitly. For instance, in [6] an operational semantics is defined while in [3] a trace based semantics is given. In both cases, the idea is to "split" the interations in the global view into its constituent send/receive actions. In this category we also put approaches like [8] where global artefacts become *global programs* with an operational semantics.

The classes above contain perfectly reasonable approaches, from a theoretical perspective. After all, we just need a semantics for the global view; whatever "fits" with the semantics of the local view would do. We argue however that making the semantics of the global view a *dependent variable* of the semantics of the local one brings in some issues that we now briefly discuss.

Firstly, several (syntactic) restrictions are usually necessary in order to rule out choreographies that "do not make sense". Such restrictions may be innocuous (as for instance the requirement that the components involved in two sequentially consecutive interactions cannot be disjoint), but they could also limit the expressiveness of the language at hand (for instance, languages featuring the parallel composition of global artefacts do not allow components involved more than one parallel thread).

Secondly, and more crucially, the semantics of global views proposed so far appear to be "too concrete". As a matter of fact, this spoils the beauty of the interplay between global and local views. All the semantics of the global view that we are aware of basically mirror quite closely the one of the local view. This means that to understand a global artefact one has to look at (or think in terms of) the corresponding local artefacts. This is not only difficult to do, but also undesirable. For instance, designers have to know/fix low level details at early stages of the development and cannot really compare different global artefacts with each other without considering the local artefacts; this makes it hard to e.g., take design decisions at the abstract level.

So, what about giving a semantics of the global view *independently* of the one of the local view? This is what we do here. We define a new semantics of global views that does not make assumptions on how messages are exchanged at lower levels. Conceptually this is easy to achieve. We fix a specification language of global artefacts and we interpret a specification as a set of "minimal and natural" causal dependencies among the messages. We then define when a global artefact is sound, namely when its causal dependencies are consistent so that they are amenable to be executed distributively by some local artefacts, regardless of the underlying message passing semantics.

We illustrate the advantages of our approach by adopting a rather liberal language of global artefacts inspired by global graphs [9]. We then show the relation of such language on a local view featuring local artefacts as communicating machines [4].

3 Global views as Graphs

Let P be a set of *participants* (ranged over by A, B, etc.), M a set of *messages* (ranged over by m, x, etc.), and K a set of *control points* (ranged over by i, j, etc.). We take P, M, and K pairwise disjoint.

The participants of a choreography exchange messages to coordinate with each other. In the global view this is modelled with *interactions* $\mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}$, which represent the fact that participant A sends message m to participant B, which is expected to receive m. A *global choreography* (g-choreography for short) is a term G derived by the following grammar (recursion is omitted for simplicity as discussed in Section 8)

$$\mathsf{G} \;::=\; \mathbf{0} \;\big|\; \mathsf{i}\colon \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B} \;\big|\; \mathsf{G};\mathsf{G}' \;\big|\; \mathsf{i}\colon(\mathsf{G}|\mathsf{G}') \;\big|\; \mathsf{i}\colon(\mathsf{G}+\mathsf{G}') \tag{2}$$

We take g-choreographies up to the structural congruence relation induced by the following axioms:

- $\_+\_$ and $\_|\_$ form a commutative monoid wrt $\mathbf{0}$
- $\_;\_$ is associative, and $\mathsf{G};\mathbf{0} = \mathsf{G}$, and $\mathbf{0};\mathsf{G} = \mathsf{G}$

A g-choreography can be empty, a simple interaction, the sequential or parallel composition of g-choreographies, or the choice between two g-choreographies. We silently assume $\mathsf{A} \neq \mathsf{B}$ in interactions $\mathsf{i}\colon \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}$. In (2), a *control point* i tags interaction, choice, and parallel g-choreographies: we assume that in a g-choreography G any two control points occurring in different positions are different, e.g., we cannot write $\mathsf{i}\colon(\mathsf{j}\colon \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}|\mathsf{i}\colon \mathsf{C} \xrightarrow{\mathsf{y}} \mathsf{D})$. Control points are a convenient technical device (as we will see when defining projections and semantics of g-choreographies) and they could be avoided.[4] Let $\mathscr{G}$ be the set of g-choreographies and, for $\mathsf{G} \in \mathscr{G}$, let $\mathsf{cp}(\mathsf{G})$ denote the set of control points in G. Through the paper we may omit control points when immaterial, e.g., writing $\mathsf{G}+\mathsf{G}'$ instead of $\mathsf{i}\colon(\mathsf{G}+\mathsf{G}')$. Finally, fix a function $\mu : \mathscr{G} \to (\mathsf{K} \to \mathsf{K})$ such that for all $\mathsf{G} \in \mathscr{G}$ $\mu(\mathsf{G})$ (written μ_{G})

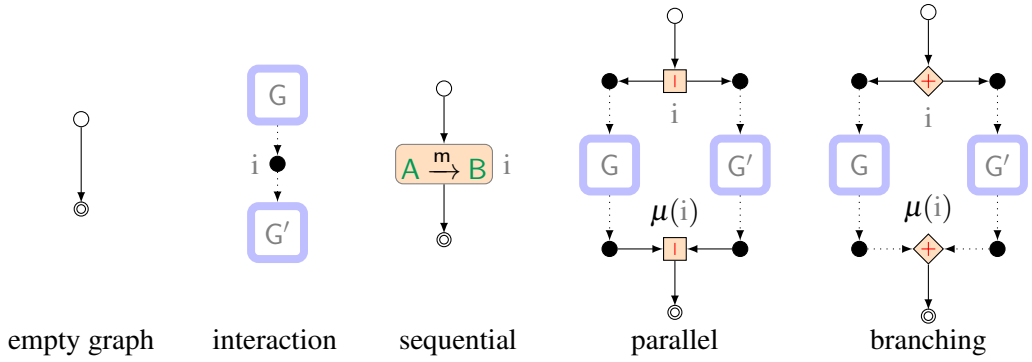
- is bijective when restricted to $\mathsf{cp}(\mathsf{G})$ and
- for all $\mathsf{i} \in \mathsf{cp}(\mathsf{G})$, $\mu_{\mathsf{G}}(\mathsf{i}) \notin \mathsf{cp}(\mathsf{G})$.

As clear in Section 5 (where we map g-choreographies on hypergraphs), μ will be used to establish a bijective relation between fork and merge control points corresponding to choices (and, in Section 4, for a bijective correspondence between (control points of) complementary send/receive actions).

The syntax in (2) captures the structure of a visual language of directed acyclic graphs[5] so that each g-choreography G can be represented as a rooted graph with a single "enter" ("exit") control point; that is G has a distinguished *source* (resp. *sink*) control point that can reach (resp. be reached by) any other control point in G. Figure 1 illustrate this. There the dotted edges from/to a $\bullet$-control points single out the source/sink control point of the graph the edge connects to. For instance, in the graph for the sequential composition, one coalesces the sink control point of G with the source control point of G', so the dotted edge from G is the incoming edge of the sink control point of G; likewise, the dotted edge from $\bullet$ is the unique outgoing edge of the source of G'. In a graph $\mathsf{G} \in \mathscr{G}$, to each node i of a branch/fork corresponds the node $\mu_{\mathsf{G}}(\mathsf{i})$ of its control point. Labels will not be depicted when immaterial. Our graphs resemble the global graphs of [9, 12] the only differences being that

- by construction, forking and branching control points i have a corresponding join and merge control point $\mu(\mathsf{i})$;
- there is a unique sink control point with a unique incoming edge (as in [9, 12], there is also a unique source control point with a unique outgoing edge).

[4] At the cost of adding technical complexity, one can automatically assign a unique identifier to such control points.
[5] Cycles are not considered for simplicity and can be easily added.



empty graph interaction sequential parallel branching

Figure 1: Our graphs: ○ is the source node, ◎ tge sink one; other nodes are drawn as ●

4 Hypergraphs of events

The semantics of a choice-free g-choreography $\mathsf{G} \in \mathscr{G}$ (i.e. a choreography that does not contain $\_+\_$ terms) is a partial order, which represents the causal dependencies of the communication actions specified by G. Choices are a bit more tricky. Intuitively, the semantics of $\mathsf{i} : (\mathsf{G} + \mathsf{G}')$ consists of two partial orders, one representing the causal dependencies of the communication actions of G and the other of those of G'. In the following, we will use hypergraphs as a compact representations of sets of partial orders.

Actions "happen" on *channels*, which we identify by the names of the participants involved in the communication. More precisely, a channel is an element of the set $\mathsf{C} = \mathsf{P}^2 \setminus \{(\mathsf{A},\mathsf{A}) \mid \mathsf{A} \in \mathsf{P}\}$ and abbreviate $(\mathsf{A},\mathsf{B}) \in \mathsf{C}$ as AB. We define some auxiliary operations first.

The set of *events* E (ranged over by e, e', f, etc.) is defined by $\mathsf{E} = \mathsf{E}^! \cup \mathsf{E}^? \cup \mathsf{K}$ where

$$\mathsf{E}^! = \mathsf{C} \times \{!\} \times \mathsf{K} \times \mathsf{M} \qquad \text{and} \qquad \mathsf{E}^? = \mathsf{C} \times \{?\} \times \mathsf{K} \times \mathsf{M}$$

are respectively the output and the input events; we shorten $(\mathsf{AB},!,\mathsf{i},\mathsf{m})$ as $\mathsf{AB}!^{\mathsf{i}}\mathsf{m}$ and $(\mathsf{AB},?,\mathsf{i},\mathsf{m})$ as $\mathsf{AB}?^{\mathsf{i}}\mathsf{m}$. The former represent *sending* actions $\mathsf{AB}!^{\mathsf{i}}\mathsf{m}$ and the latter represents *receiving* actions $\mathsf{AB}?^{\mathsf{i}}\mathsf{m}$; the *subject* of the actions is $\mathsf{sbj}(\mathsf{AB}!^{\mathsf{i}}\mathsf{m}) = \mathsf{A}$ (i.e., the sender) and $\mathsf{sbj}(\mathsf{AB}?^{\mathsf{i}}\mathsf{m}) = \mathsf{B}$ (i.e., the receiver), respectively. As will be clear later, events in K represent "non-observable" actions like (the execution of) a choice or a merge; we assume that $\mathsf{sbj}(\_)$ is undefined on K.

The communication action of e is defined by: $\mathsf{act}(\mathsf{AB}?^{\mathsf{i}}\mathsf{m}) = \mathsf{AB}?\mathsf{m}$ and $\mathsf{act}(\mathsf{AB}!^{\mathsf{i}}\mathsf{m}) = \mathsf{AB}!\mathsf{m}$ and we extend $\mathsf{cp}()$ to events, so $\mathsf{cp}(e)$ denotes the control point of an event e. When considering sets of events $\tilde{e} \in 2^{\mathsf{E}}$, we will tacitly assume that any two events have different control points (that is for all $e, e' \in \tilde{e}$, $\mathsf{cp}(e) \neq \mathsf{cp}(e')$). Also, we write $e \in \mathsf{G}$ when there is an interaction $\mathsf{i} : \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}$ in G such that $e \in \{\mathsf{AB}!^{\mathsf{i}}\mathsf{m}, \mathsf{AB}?^{\mathsf{i}}\mathsf{m}\}$, and accordingly $\tilde{e} \subseteq \mathsf{G}$ means that $e \in \mathsf{G}$ for all $e \in \tilde{e}$.

A relation $R \subseteq 2^{\mathsf{E}} \times 2^{\mathsf{E}}$ on sets of events is a directed hypergraph, that is a graph where nodes are events and hyperarcs $(\!|\tilde{e},\tilde{e}'|\!)$ relate sets of events, the source $\tilde{e}$ and the target $\tilde{e}'$. Let $\widehat{R} = \{\langle e, e'\rangle \in \mathsf{E} \times \mathsf{E} \mid \exists (\!|\tilde{e},\tilde{e}'|\!) \in R : e \in \tilde{e} \text{ and } e' \in \tilde{e}'\} \subseteq \mathsf{E} \times \mathsf{E}$ be the *happen-before* relation induced by R, namely $\langle e, e'\rangle \in R$ when e precedes e' in R. Intuitively, $\widehat{R}$ are the causal dependencies R among the events in R. To avoid cumbersome parenthesis, singlelton sets in hyperarcs are shortened by their element (e.g., we write $(\!|e,\tilde{e}|\!)$ instead of $(\!|\{e\},\tilde{e}|\!)$). Let $\pi_1 : 2^{\mathsf{E}} \times 2^{\mathsf{E}} \to 2^{\mathsf{E}}$ be such that $\pi_1((\!|\tilde{e},\tilde{e}'|\!)) = \tilde{e}$.

Given $R, R' \subseteq 2^{\mathsf{E}} \times 2^{\mathsf{E}}$, define the hypergraphs $R \circ R'$ and R^* respectively as

$$R \circ R' = \{(\!|\tilde{e},\tilde{e}'|\!) \mid \exists (\!|\tilde{e},\tilde{e}_1|\!) \in R, (\!|\tilde{e}_2,\tilde{e}'|\!) \in R' : \tilde{e}_1 \cap \tilde{e}_2 \neq \emptyset\} \qquad \text{and} \qquad R^* = \bigcup_n \underbrace{R \circ \cdots \circ R}_{n\text{-times}}$$

Basically, R^* is the reflexo-transitive closure of R with respect to the composition relation $\circ$.

We define the maximal and minimal elements of R respectively as

$$\max R = \{e \in \mathsf{E} \mid \nexists (\!|\tilde{e}, \tilde{e}'|\!) \in R \wedge e \in \tilde{e}\} \qquad \text{and} \qquad \min R = \{e \in \mathsf{E} \mid \nexists (\!|\tilde{e}, \tilde{e}'|\!) \in R \wedge e \in \tilde{e}'\}$$

We also need to define the (hyperedges insisting on) "last" and the "first" communication actions in R.

$$\mathsf{lst}\, R = \{(\!|\tilde{e}, \tilde{e}'|\!) \in R \mid \tilde{e}' \cap \mathsf{K} = \emptyset \wedge \forall (\!|\tilde{e}', \tilde{e}''|\!) \in R^\star : \tilde{e}'' \subseteq \mathsf{K}\} \qquad \text{and} \qquad \mathsf{fst}\, R = \left(\mathsf{lst}\,(R^{-1})\right)^{-1}$$

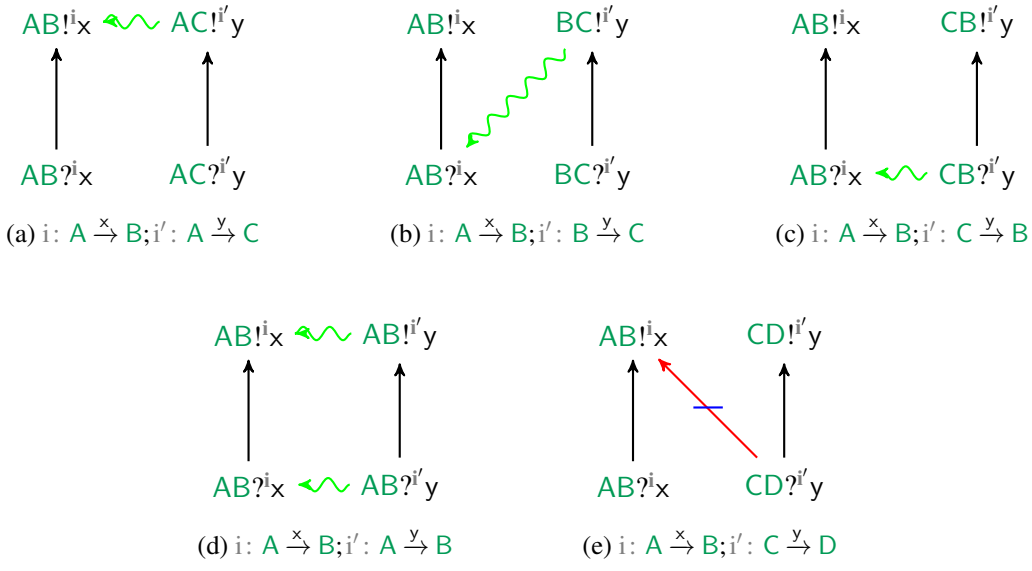
The operations above are instrumental to define the "sequential " composition of relations R and R' on E as follows:

$$\begin{aligned} \mathsf{seq}(R, R') \;=\; & R \cup R' \\ \cup\; & \left\{(\!|e, e'|\!) \in (\mathsf{E} \setminus \mathsf{K})^2 \mid \exists (\!|\tilde{e}_1, \tilde{e}_2|\!) \in \mathsf{lst}\, R, (\!|\tilde{e}'_1, \tilde{e}'_2|\!) \in \mathsf{fst}\, R' : \right. \\ & \left. e \in (\tilde{e}_1 \cup \tilde{e}_2) \setminus \mathsf{K} \;\wedge\; e' \in (\tilde{e}'_1 \cup \tilde{e}'_2) \setminus \mathsf{K} \;\wedge\; \mathsf{sbj}(e) = \mathsf{sbj}(e')\right\} \end{aligned}$$

The sequential composition of two hypergraphs R and R' preserves the causal dependencies of its constituents, namely those in $R \cup R'$. Additionally, dependencies are established between every event in $\mathsf{lst}\, R$ and every event in $\mathsf{fst}\, R'$ that have the same subject. Fig. 2 depicts the sequential compositions of two hypergraphs, say R and R'. The former hypergraph corresponds to the interaction $\mathrm{i} : \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}$, while the second corresponds to the interactions

$$\mathrm{i}' : \mathsf{A} \xrightarrow{\mathsf{y}} \mathsf{C} \qquad \mathrm{i}' : \mathsf{B} \xrightarrow{\mathsf{y}} \mathsf{C} \qquad \mathrm{i}' : \mathsf{C} \xrightarrow{\mathsf{y}} \mathsf{B} \qquad \mathrm{i}' : \mathsf{A} \xrightarrow{\mathsf{y}} \mathsf{B} \qquad \mathrm{i}' : \mathsf{C} \xrightarrow{\mathsf{y}} \mathsf{D}$$

in each case respectively. In Fig. 2, the events on control point i belongs to R while those on control point i' to R'; also, simple arrows represent the dependencies induced by the subjects and wavy arrows represent dependencies induced by the sequential composition; the meaning of stroken arrows will be explained in Section 5.



(a) $\mathrm{i} : \mathsf{A} \xrightarrow{\mathsf{x}} \mathsf{B}; \mathrm{i}' : \mathsf{A} \xrightarrow{\mathsf{y}} \mathsf{C}$ (b) $\mathrm{i} : \mathsf{A} \xrightarrow{\mathsf{x}} \mathsf{B}; \mathrm{i}' : \mathsf{B} \xrightarrow{\mathsf{y}} \mathsf{C}$ (c) $\mathrm{i} : \mathsf{A} \xrightarrow{\mathsf{x}} \mathsf{B}; \mathrm{i}' : \mathsf{C} \xrightarrow{\mathsf{y}} \mathsf{B}$

(d) $\mathrm{i} : \mathsf{A} \xrightarrow{\mathsf{x}} \mathsf{B}; \mathrm{i}' : \mathsf{A} \xrightarrow{\mathsf{y}} \mathsf{B}$ (e) $\mathrm{i} : \mathsf{A} \xrightarrow{\mathsf{x}} \mathsf{B}; \mathrm{i}' : \mathsf{C} \xrightarrow{\mathsf{y}} \mathsf{D}$

Figure 2: Examples of sequential composition

We now define the concept of "common" part of two hypergraphs R and R' with respect to a participant A. Given a set of events $\tilde{e}$ in R and one $\tilde{e}'$ in R', we say that $\tilde{e}'$ A-*reflects* $\tilde{e}$ if, and only if, there is a bijection $f_{\mathsf{A}} : \tilde{e} \to \tilde{e}'$ such that the following conditions hold

- $\forall e \in \tilde{e} : \mathsf{sbj}(\mathsf{e}) = \mathsf{sbj}(\mathsf{f_A}(\mathsf{e})) = \mathsf{A} \wedge \mathsf{act}(e) = \mathsf{act}(f_{\mathsf{A}}(e))$
- $\forall e' \in \tilde{e}\ \forall \langle e, e' \rangle \in \widehat{R} : \mathsf{sbj}(\mathsf{e}) = \mathsf{A} \implies \big(e \in \tilde{e} \wedge \langle f_{\mathsf{A}}(e), f_{\mathsf{A}}(e') \rangle \in \widehat{R'}\big)$
- $\forall e' \in f_{\mathsf{A}}(\tilde{e})\ \forall \langle e, e' \rangle \in \widehat{R'} : \mathsf{sbj}(\mathsf{e}) = \mathsf{A} \implies \big(e \in f_{\mathsf{A}}(\tilde{e}) \wedge \langle f_{\mathsf{A}}^{-1}(e), f_{\mathsf{A}}^{-1}(e') \rangle \in \widehat{R}\big)$.

The A-*only* part of a set of events $\tilde{e} \in 2^{\mathsf{E}}$ is the set $\tilde{e}^{@\mathsf{A}}$ where the actions of $\tilde{e}$ not having subject A are replaced with the control point of the action; formally

$$\begin{array}{rcl} \tilde{e}^{@\mathsf{A}} & = & \{e \in \tilde{e} \mid \mathsf{sbj}(\mathsf{e}) = \mathsf{A} \vee e \in \mathsf{K}\} \\ & \cup & \{\mathsf{cp}(e) \mid e \in \tilde{e} \cap \mathsf{E}^{!} \wedge \mathsf{sbj}(\mathsf{e}) \neq \mathsf{A}\} \cup \{\mu(\mathsf{cp}(\mathsf{e})) \mid e \in \tilde{e} \cap \mathsf{E}^{?} \wedge \mathsf{sbj}(\mathsf{e}) \neq \mathsf{A}\} \end{array}$$

Accordingly, the A-*only* part of a hypergraphs R is defined as $R^{@\mathsf{A}} = \big\{ (\!| \tilde{e}^{@\mathsf{A}}, \tilde{e}'^{@\mathsf{A}} |\!) \bigm| (\!| \tilde{e}, \tilde{e}' |\!) \in R \big\}$. Notice that we use $\mathsf{cp}(e)$ and $\mu(\mathsf{cp}(\mathsf{e}))$ for outputs and inputs respectively, so that different events not belonging to A remain distinguished.

The notion of reflections is new. It will allow us to define *active* and *passive* participants in a choice.

5 Semantics of Choreographies

The semantics of g-choreography is the partial map $[\![\_]\!]_\mu : \mathscr{G} \to 2^{(2^{\mathsf{E}} \times 2^{\mathsf{E}})}$ defined[6] as:

$$[\![\mathbf{0}]\!] = \emptyset \tag{3}$$

$$[\![\mathsf{i} : \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}]\!] = \{(\!| \mathsf{AB}!^{\mathsf{i}}\mathsf{m}, \mathsf{AB}?^{\mathsf{i}}\mathsf{m} |\!)\} \tag{4}$$

$$[\![\mathsf{i} : (\mathsf{G} | \mathsf{G}')]\!] = [\![\mathsf{G}]\!] \cup [\![\mathsf{G}']\!] \tag{5}$$

$$[\![\mathsf{i} : \mathsf{G}; \mathsf{G}']\!] = \begin{cases} R & \text{if } R = \mathsf{seq}([\![\mathsf{G}]\!], [\![\mathsf{G}']\!]) \text{ and } R^* \supseteq \pi_1(\mathsf{lst}\,[\![\mathsf{G}]\!]) \times \pi_1(\mathsf{fst}\,[\![\mathsf{G}']\!]) \\ \bot & \text{otherwise} \end{cases} \tag{6}$$

$$[\![\mathsf{i} : (\mathsf{G} + \mathsf{G}')]\!] = \begin{cases} [\![\mathsf{G}]\!] \cup [\![\mathsf{G}']\!] \cup R & \text{if } R = \{(\!| \mathsf{i}, \min [\![\mathsf{G}]\!] |\!), (\!| \mathsf{i}, \min [\![\mathsf{G}']\!] |\!), (\!| \max [\![\mathsf{G}]\!], \mathsf{j} |\!), (\!| \max [\![\mathsf{G}']\!], \mathsf{j} |\!)\} \\ & \quad \text{and } \mathsf{j} = \mu(\mathsf{i}) \text{ and } wb(\mathsf{G}, \mathsf{G}') \\ \bot & \text{otherwise} \end{cases} \tag{7}$$

The semantics of the the empty g-choreography $\mathbf{0}$ and of interaction $\mathsf{i} : \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}$ are straightforward; for the latter, the send part $\mathsf{AB}!^{\mathsf{i}}\mathsf{m}$ of the interaction must precede its receive $\mathsf{AB}?^{\mathsf{i}}\mathsf{m}$ part.

For the parallel composition $\mathsf{i} : (\mathsf{G} | \mathsf{G}')$ we just take the union of the dependencies of G and G', thus allowing the arbitrary interleaved of those events.

The semantics of sequential composition $\mathsf{i} : \mathsf{G}; \mathsf{G}'$ establishes happens before relations as computed by $\mathsf{seq}([\![\mathsf{G}]\!], [\![\mathsf{G}']\!])$ provided that they cover all the dependencies between all last actions of G with all first actions of G', ensuring the soundness of the composition. If this condition does not hold, then there is a participant in G' that cannot ascertain if all the events of G did happen before it could start. All examples Fig. 2 are sound, barred the one in Fig. 2e, where the stroken edge depicts the missing dependency that is not guaranteed by the hypergraph.

[6] We assume μ to be understood and simply write $[\![\_]\!]$.

The semantics of a choice $\mathsf{i}:(\mathsf{G}+\mathsf{G}')$ is defined provided that the *well-branched* condition $wb(\mathsf{G},\mathsf{G}')$ holds on G and G', that is when (*i*) there is at most one *active* participant and (*ii*) all the other participants are *passive*. In a moment, after some auxiliary definitions, we define active and passive participants. Intuitively, the notions of active and passive participant (formalised in the following, single out those participants A that do not make an internal choice, namely it is not A selecting whether to execute G or G' and those instead that perform an internal choice selecting which branch to execute. Besides the dependencies induced by G and G', $[\![\mathsf{i}:(\mathsf{G}+\mathsf{G}')]\!]$ contains those making i (the control point of the branch) precede every minimal events of G and G'; similarly, the maximal events of G and G' have to precede the merge control point $\mu(\mathsf{i})$ (which marks the conclusion of the choice). Notice that no additional dependency is added between the events of the constituents. In fact, during one instance of the g-choreography either the actions of the first branch or the actions of the second one will be performed.

Auxiliary definitions The relation $<_{\mathsf{G}}$ is the *happens-before relation induced by* $\mathsf{G}\in\mathscr{G}$ defined as $<_{\mathsf{G}}=\widehat{([\![\mathsf{G}]\!]^*)}$ if $[\![\mathsf{G}]\!]$ is defined, and $<_{\mathsf{G}}=\emptyset$ otherwise. Notice that $<_{\mathsf{G}}$ is a partial order on the events of G. For $R\subseteq 2^{\mathsf{E}}\times 2^{\mathsf{E}}$ and $\tilde{e}\subseteq 2^{\mathsf{E}}$, let $R\setminus\tilde{e}=\{(\!|\tilde{e}_1\setminus\tilde{e},\tilde{e}_2\setminus\tilde{e}|\!) \mid (\!|\tilde{e}_1,\tilde{e}_2|\!)\in R\}$. Given a participant $\mathsf{A}\in\mathsf{P}$, two g-choreographies $\mathsf{G},\mathsf{G}'\in\mathscr{G}$, and two set of events $\tilde{e}\subseteq\mathsf{G}$ and $\tilde{e}'\subseteq\mathsf{G}'$ such that $\tilde{e}'$ A-reflects $\tilde{e}$, define $\mathsf{div}_{\mathsf{A}}^{\tilde{e},\tilde{e}'}(\mathsf{G},\mathsf{G}')=(\tilde{e}_1,\tilde{e}_2)$ where

$$\tilde{e}_1 \;=\; \bigcup\pi_1(\mathsf{fst}([\![\mathsf{G}]\!]^{@\mathsf{A}})\setminus\tilde{e}) \qquad\qquad \tilde{e}_2 \;=\; \bigcup\pi_1(\mathsf{fst}([\![\mathsf{G}']\!]^{@\mathsf{A}})\setminus\tilde{e}')$$

is the A-*branching pair* in choice $\mathsf{G}+\mathsf{G}'$ with respect to $\tilde{e}$ and $\tilde{e}'$. Intuitively, the behavior of A in the two branches G and G' can be the same up to her point of branching $\mathsf{div}_{\mathsf{A}}^{\tilde{e},\tilde{e}'}(\mathsf{G},\mathsf{G}')$. The A-*reflectivity* is used to identify such common behavior (i.e. all events in $\tilde{e}$ and $\tilde{e}'$) and to ignore it when checking the behaviour of A in the branches. In fact, by taking the A-only parts of these hypergraphs and selecting their fist interactions (that is the components $\tilde{e}_1$ and $\tilde{e}_2$ of the A-branching pair) we identify when the behavior of A in G starts to be different from her behaviour in G'.

Active and passive roles We use $\sqcap$ to represent the intersection of sets of events disregarding the control points; formally $\tilde{e}\sqcap\tilde{e}'=act(e):e\in\tilde{e}\cap act(e'):e'\in\tilde{e}'$. A participant $\mathsf{A}\in\mathsf{P}$ is *passive* in $\mathsf{G}+\mathsf{G}'$ respect $\tilde{e}$ and $\tilde{e}'$ if let $(\tilde{e}_1,\tilde{e}_2)=\mathsf{div}_{\mathsf{A}}^{\tilde{e},\tilde{e}'}(\mathsf{G},\mathsf{G}')$ then

$$\begin{array}{l|l} \tilde{e}_1\sqcap\{e\in\mathsf{G}' \mid \nexists e'\in\tilde{e}_2 : e<_{\mathsf{G}'}e'\}=\emptyset & \tilde{e}_1\cup\tilde{e}_2\subseteq\mathsf{E}^? \\ \tilde{e}_2\sqcap\{e\in\mathsf{G} \mid \nexists e'\in\tilde{e}_1 : e<_{\mathsf{G}}e'\}=\emptyset & \tilde{e}_1=\emptyset\iff\tilde{e}_2=\emptyset \end{array}$$

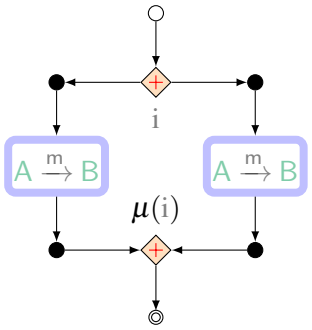
Thus, the behavior of A in G and G' must be the same up to a point where she receives either of two different messages, each one identifying which branch had been selected. Clearly, A cannot perform outputs at the points of divergence. We say that a participant A is *passive* in $\mathsf{G}+\mathsf{G}'$ if such $\tilde{e}$ and $\tilde{e}'$ exist.

A participant $\mathsf{A}\in\mathsf{P}$ is *active* in $\mathsf{G}+\mathsf{G}'$ respect $\tilde{e}$ and $\tilde{e}'$ if let $(\tilde{e}_1,\tilde{e}_2)=\mathsf{div}_{\mathsf{A}}^{\tilde{e},\tilde{e}'}(\mathsf{G},\mathsf{G}')$ then

$$\tilde{e}_1\cup\tilde{e}_2\subseteq\mathsf{E}^! \qquad \tilde{e}_1\sqcap\tilde{e}_2=\emptyset \qquad \tilde{e}_1\neq\emptyset \qquad \tilde{e}_2\neq\emptyset$$

Thus, the behavior of A in G and G' must be the same up to the point where she informs the other participants, by sending different messages, which branch she chose. We say that a participant A is *active* in $\mathsf{G}+\mathsf{G}'$ if such $\tilde{e}$ and $\tilde{e}'$ exist. (The active participant of a choice is sometimes called *selector*.) Interestingly, if one takes the empty reflection in the determination of active and passive roles, the definition above yield exactly the same notions used e.g., in [10, 3, 7].

Some examples Unlike its corresponding notions in the rest of the literature, well-branchedness does not require the selector to exist. For instance, the choreography



$$= \quad \mathsf{i} : (\mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B} + \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B})$$

is well-branched even if there is not active (i.e., selector) participant. We are not aware of any other framework where this is the case.

The hypergraphs in Fig. 3 are respectively the semantics of the g-choreographies

$$\mathsf{G}_{(3a)} = \mathsf{i}_3 : (\mathsf{i}_1 : \mathsf{A} \xrightarrow{\mathsf{x}} \mathsf{B} + \mathsf{i}_2 : \mathsf{A} \xrightarrow{\mathsf{y}} \mathsf{B}) \tag{8}$$

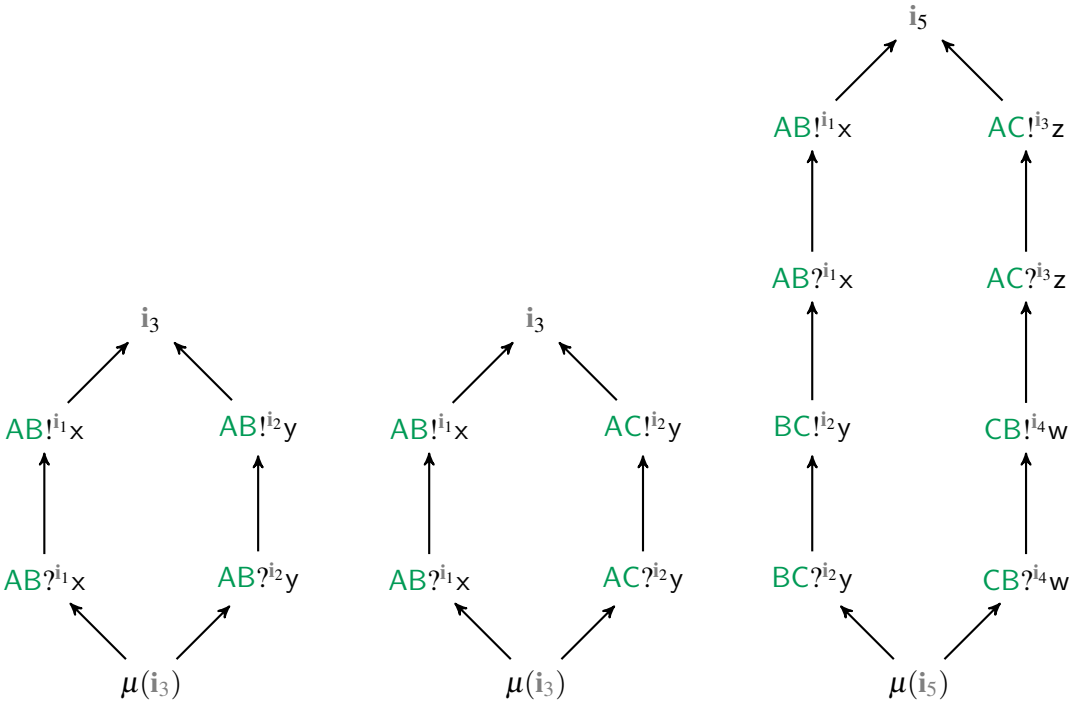
$$\mathsf{G}_{(3b)} = \mathsf{i}_3 : (\mathsf{i}_1 : \mathsf{A} \xrightarrow{\mathsf{x}} \mathsf{B} + \mathsf{i}_2 : \mathsf{A} \xrightarrow{\mathsf{y}} \mathsf{C}) \tag{9}$$

$$\mathsf{G}_{(3c)} = \mathsf{i}_5 : (\left(\mathsf{i}_1 : \mathsf{A} \xrightarrow{\mathsf{x}} \mathsf{B}; \mathsf{i}_2 : \mathsf{B} \xrightarrow{\mathsf{y}} \mathsf{C}\right) + \left(\mathsf{i}_3 : \mathsf{A} \xrightarrow{\mathsf{z}} \mathsf{C}; \mathsf{i}_4 : \mathsf{C} \xrightarrow{\mathsf{w}} \mathsf{B}\right)) \tag{10}$$

Figure 3a the choice is well-branched; the participant B is passive (receiving either AB?x or AB?y in the point of divergence). and the participant A is active (sending either AB!x or AB!y in the point of divergence).
Figure 3b the choice is not well-branched; the participant A is active (sending either AB!x or AC!y in the point of divergence), however, B (and C) is neither passive or active (in one branch the events of divergence is AB?x while for the other branch it is empty).
Figure 3c the choice is well-branched; A is active (sending either AB!x or AC!z in the point of divergence), B is passive (it receives either AB?x or CB?w in the events of divergence), and C is passive (it receives either BC?y or AC?z in the events of divergence).
Figure 4 the choice is well-branched; A is active (it has the same behavior in the branches i_3 and i_6, so its events of divergence are AC!z and AC!w), B is passive (it has the same behavior in the branches i_3 and i_6 and its events of divergence are empty), and C is passive (it receives either AC?z or AC?w in the events of divergence).

6 Languages of Choreographies

The abstract semantics of a g-choreography is a hypergraph, which represents the set of partial orders among the events of the g-choreography. A more concrete semantics can be given by considering the *language* of a g-choreography. Informally, the language of a g-choreography $\mathsf{G} \in \mathscr{G}$ is the sequences of words made of the communication actions of the events in G that preserve the causal relations of $[\![\mathsf{G}]\!]$, provided that $[\![\mathsf{G}]\!]$ is defined.



(a) $\mathsf{G}_{(3a)}$ (b) $\mathsf{G}_{(3b)}$ (c) $\mathsf{G}_{(3c)}$

Figure 3: Some examples

Given a g-choreography G, let $\mathsf{G}^{\oplus} = [\![\mathsf{G}]\!] \cap (2^{\mathsf{K}} \times 2^{\mathsf{E}})$ be the set of *choice hyperedges* of G (that is those hyperedges in G whose source represents choices) and define the outgoing hyperedges of $\mathrm{i} \in \mathsf{K}$ in G as $\mathsf{G}^{\oplus}(\mathrm{i}) = \mathsf{G}^{\oplus} \cap (\{\{\mathrm{i}\}\} \times 2^{\mathsf{E}})$. A map $c : \mathsf{G}^{\oplus} \to 2^{\mathsf{E}}$ is a *resolution of* G if $c(\mathrm{i}) \in \mathsf{G}^{\oplus}(\mathrm{i})$ for every $\mathrm{i} \in \mathsf{K}$. Intuitively, a resolution fixes a branch for every choice in a g-choreography G and therefore it induces a preorder of the events compatible with G and the resolution.

The preorder corresponding to a resolution is computed by $\mathsf{G}\circledR c$. This hypergraph is obtained by (*i*) removing every hyperedge not chosen by the resolution and (*ii*) removing every dead event (e.g. events that are note reachable from the initial events after removing the non-selected hyperedges):

$$\mathsf{G}\circledR c \quad = \quad (\mathit{trim}([\![\mathsf{G}]\!] \setminus \bigcup_{\mathrm{i} \in \mathsf{G}^{\oplus}} (\mathsf{G}^{\oplus}(\mathrm{i}) \setminus c(\mathrm{i})), \min [\![\mathsf{G}]\!]))$$

where $\mathit{trim}(R, \tilde{e})$ is the function that remove every node in the hypergraph R that is not reachable from $\tilde{e}$.

Let $\mathscr{A} = \mathsf{E}^{!} \cup \mathsf{E}^{?}$. The *language* of a g-choreography $\mathsf{G} \in \mathscr{G}$ is

$$\mathscr{L}[\mathsf{G}] = \{\mathsf{act}(w) \mid w \in \mathscr{A}^{\star} \text{ and } \exists \text{ a resolution } c \text{ of } \mathsf{G} : \psi(w, c)\}$$

where, $\psi(w, c)$ holds iff for each indexes $i \neq j$ between 1 and the size of w we have that

1. $w[i] \neq w[j]$, where $w[i]$ stands for the i-th symbol in w
2. $w[i] \in \mathsf{G}\circledR c$
3. if $w[i] <_{\mathsf{G}\circledR c} w[j]$ then $i < j$
4. for every e, if $e <_{\mathsf{G}\circledR c} w[i]$ then there exists $h < i$ such that $w[h] = e$

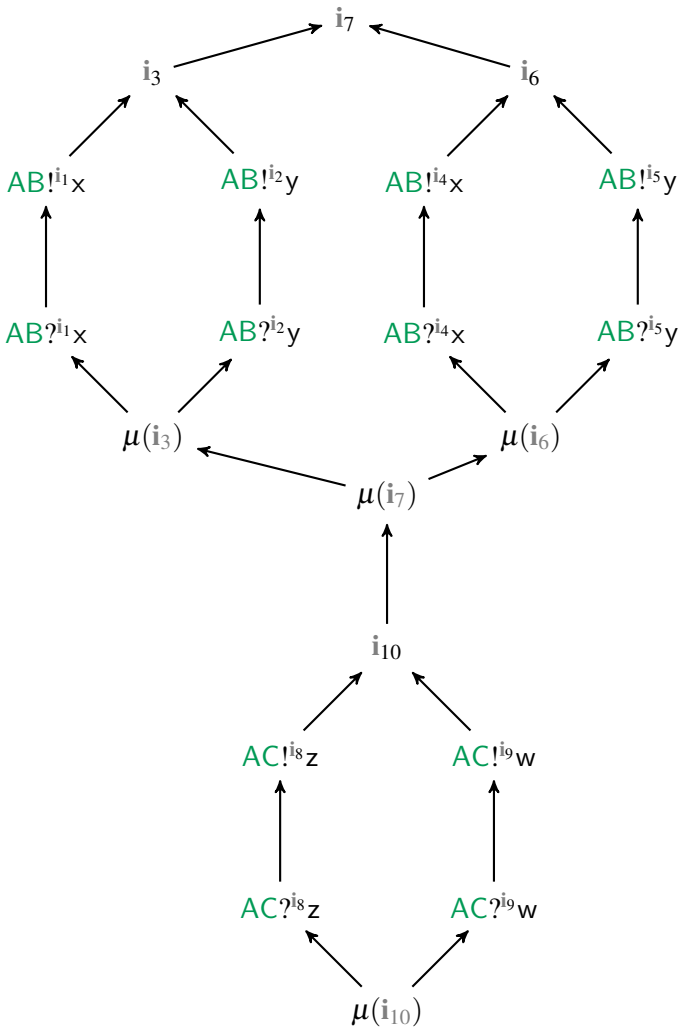


Figure 4: $\mathbf{i}_7 : (\mathbf{i}_3 : (\mathbf{i}_1 : \mathsf{A} \xrightarrow{\mathsf{x}} \mathsf{B} + \mathbf{i}_2 : \mathsf{A} \xrightarrow{\mathsf{y}} \mathsf{B}) + \mathbf{i}_6 : (\mathbf{i}_4 : \mathsf{A} \xrightarrow{\mathsf{x}} \mathsf{B} + \mathbf{i}_5 : \mathsf{A} \xrightarrow{\mathsf{y}} \mathsf{B})); \mathbf{i}_{10} : (\mathbf{i}_8 : \mathsf{A} \xrightarrow{\mathsf{z}} \mathsf{C} + \mathbf{i}_9 : \mathsf{A} \xrightarrow{\mathsf{w}} \mathsf{C})$

Items 1 and 2 state that events in the word are not repeated and that the word is made only of events present in the preorder, i.e. the word can not mix events belonging to two different branches. Item 3 states that words preserve the causal relations of events. Item 4 requires that all the predecessors of an event in the word must precede the event in the word. Notice that $\mathscr{L}[\mathsf{G}]$ is the prefix-closed.

7 Projecting on Communicating Machines

As in [12, 9], we adopt *communicating finite state machines* (CFSM) as local artefacts. We borrow the definition of CFSMs in [4], with slight adaptation to our context. A CFSM is a finite transition system given by a tuple $M = (Q, q_0, \rightarrow)$ where

- Q is a finite set of *states* with $q_0 \in Q$ the *initial* state, and
- $\rightarrow \subseteq Q \times \mathsf{act}(\mathscr{A}) \times Q$ is a set of *transitions*; we write $q \xrightarrow{e} q'$ for $(q, e, q') \in \rightarrow$.

A CFSM $(Q, q_0, \rightarrow)$ is A-*local* if for every $q \xrightarrow{e} q' \in \rightarrow$ holds $\mathsf{sbj}(\mathsf{e}) = \mathsf{A}$. Given a A-*local* CFSM $M_\mathsf{A} = (Q_\mathsf{A}, q_{q\mathsf{A}}, \rightarrow_\mathsf{A})$ for each $\mathsf{A} \in \mathsf{P}$, the tuple $S = (M_\mathsf{A})_{\mathsf{A}\in\mathsf{P}}$ is a *communicating system*.

The semantics of communicating systems is defined in terms of *transition systems*, which keeps track of the state of each machine and the content of each buffer. Let $S = (M_\mathsf{A})_{\mathsf{A}\in\mathsf{P}}$ be a *communicating system*. A *configuration* of S is a pair $s = \langle \tilde{q} \,;\, \tilde{b} \rangle$ where $\tilde{q} = (q_\mathsf{A})_{\mathsf{A}\in\mathsf{P}}$ with $q_\mathsf{A} \in Q_\mathsf{A}$ and where $\tilde{b} = (b_{\mathsf{AB}})_{\mathsf{AB}\in\mathsf{C}}$ with $b_{\mathsf{AB}} \in \mathsf{M}^*$; q_A keeps tracks of the state of the machine A and b_{AB} is the buffer that keeps tracks of the messages delivered from A to B. The *initial* configuration s_0 is the one where q_A is the initial state of the corresponding CFSM and all buffers are empty.

A configuration $s' = \langle \tilde{q}' \,;\, \tilde{b}' \rangle$ is *reachable* from another configuration $s = \langle \tilde{q} \,;\, \tilde{b} \rangle$ by *firing transition* e, written $s \overset{e}{\Rightarrow} s'$ if there is $\mathsf{m} \in \mathsf{M}$ such that either (1) or (2) below hold:

1. $e = \mathsf{AB}!\mathsf{m}$ and $q_\mathsf{A} \xrightarrow{e} q'_\mathsf{A} \in \rightarrow_\mathsf{A}$ and
   a. $q'_\mathsf{C} = q_\mathsf{C}$ for all $\mathsf{C} \neq \mathsf{A}$
   b. and $b'_{\mathsf{AB}} = b_{\mathsf{AB}}.\mathsf{m}$
   c. and $b'_{\mathsf{A'B'}} = b_{\mathsf{A'B'}}$ for all $(\mathsf{A'}, \mathsf{B'}) \neq (\mathsf{A}, \mathsf{B})$
2. $e = \mathsf{AB}?\mathsf{m}$ and $q_\mathsf{A} \xrightarrow{e} q'_\mathsf{A} \in \rightarrow_\mathsf{A}$ and
   a. $q'_\mathsf{C} = q_\mathsf{C}$ for all $\mathsf{C} \neq \mathsf{B}$
   b. and $b_{\mathsf{AB}} = \mathsf{m}.b'_{\mathsf{AB}}$
   c. and $b'_{\mathsf{A'B'}} = b_{\mathsf{A'B'}}$ for all $(\mathsf{A'}, \mathsf{B'}) \neq (\mathsf{A}, \mathsf{B})$

Condition (1) puts m on channel AB, while (2) gets m from channel AB.

A configuration $s = \langle \tilde{q} \,;\, \tilde{b} \rangle$ is *stable* if all buffers are empty: $\tilde{b} = \tilde{\varepsilon}$. A configuration $s = \langle \tilde{q} \,;\, \tilde{b} \rangle$ is a *deadlock* if $s \not\Rightarrow$ and

- there exists a $\mathsf{A} \in \mathsf{P}$ such that $q_\mathsf{A} \xrightarrow{\mathsf{AB}?\mathsf{m}} q'_\mathsf{A} \in \rightarrow_\mathsf{A}$
- or $\tilde{b} \neq \tilde{\varepsilon}$

The language of a communicating system S is the biggest prefix closed set $\mathscr{L}[S] \in \mathsf{act}(\mathscr{A})^\star$ such that for each $e_0 \dots e_{n-1} \in \mathscr{L}[S]$, $s_0 \overset{e_0}{\Rightarrow} \dots \overset{e_{n-1}}{\Rightarrow} s_n$.

Given two CFSMs $M = (Q, q_0, \rightarrow)$ and $M' = (Q', q'_0, \rightarrow')$, write $M \cup M'$ for the machine $(Q \cup Q', q_0, \rightarrow \cup \rightarrow')$ provided that $q_0 = q'_0$; also, $M \cap M'$ denotes $Q \cap Q'$. The product of M and M' is defined as usual as $M \times M' = (Q \times Q', (q_0, q'_0), \rightarrow'')$ where $\big((q_1, q'_1), e, (q_2, q'_2)\big) \in \rightarrow''$ if, and only if,

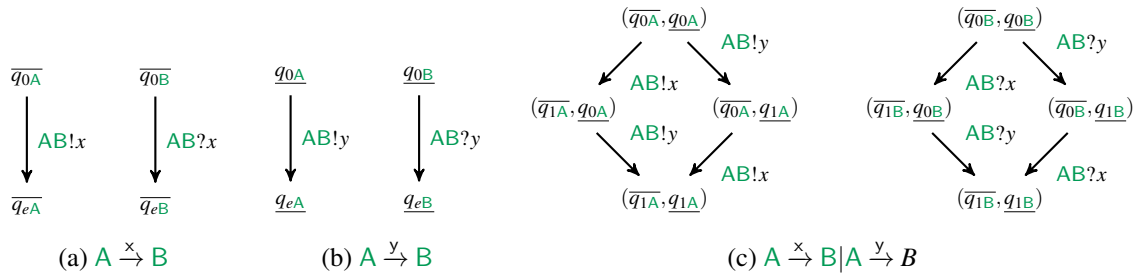
$$\big((q_1, e, q_2) \in \rightarrow \text{ and } q'_1 = q'_2\big) \qquad \text{or} \qquad \big((q'_1, e, q'_2) \in \rightarrow' \text{ and } q_1 = q_2\big)$$

We also use $min(M)$ to denote the CFSM obtained by minimising M (using e.g., the classical partition refinement algorithm) when interpreting them as finite automata.

Let G be a g-choreography, the function $\mathsf{G}\!\downarrow_\mathsf{A}$ yields the projection (in the form of a CFSM) of the choreography over the participant A using q_0 and q_e as initial and sink states respectively. The projection is defined as follow:

$$\mathsf{G}\!\downarrow_\mathsf{A}^{q_0,q_e} = \begin{cases} \rightarrow (q_0) \rightarrow & \text{if } \mathsf{G} = \mathbf{0} \text{ and } q_0 = q_e \\ \rightarrow (q_0) \rightarrow & \text{if } \mathsf{G} = \mathsf{i}\colon \mathsf{B} \xrightarrow{\mathsf{m}} \mathsf{C} \text{ and } q_0 = q_e \\ \rightarrow (q_0) \xrightarrow{\mathsf{AB}!m} (q_e) \rightarrow & \text{if } \mathsf{G} = \mathsf{i}\colon \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B} \text{ and } q_0 \neq q_e \\ \rightarrow (q_0) \xrightarrow{\mathsf{BA}?m} (q_e) \rightarrow & \text{if } \mathsf{G} = \mathsf{i}\colon \mathsf{B} \xrightarrow{\mathsf{m}} \mathsf{A} \text{ and } q_0 = q_e \\ \mathsf{G}_1\!\downarrow_\mathsf{A}^{q_0,q_e'} \cup\, \mathsf{G}_2\!\downarrow_\mathsf{A}^{q_e',q_e} & \text{if } \mathsf{G} = \mathsf{i}\colon \mathsf{G}_1;\mathsf{G}_2 \text{ and } \mathsf{G}_1\!\downarrow_\mathsf{A}^{q_0,q_e'} \cap\, \mathsf{G}_2\!\downarrow_\mathsf{A}^{q_e',q_e} = \{q_e'\} \\ \mathsf{G}_1\!\downarrow_\mathsf{A}^{q_0,q_e} \cup\, \mathsf{G}_2\!\downarrow_\mathsf{A}^{q_0,q_e} & \text{if } \mathsf{G} = \mathsf{i}\colon (\mathsf{G}_1 + \mathsf{G}_2) \text{ and } \mathsf{G}_1\!\downarrow_\mathsf{A}^{q_0,q_e} \cap\, \mathsf{G}_2\!\downarrow_\mathsf{A}^{q_0,q_e} = \{q_0, q_e\} \\ \mathsf{G}_1\!\downarrow_\mathsf{A}^{\overline{q_0},\overline{q_e}} \times \mathsf{G}_2\!\downarrow_\mathsf{A}^{\underline{q_0},\underline{q_e}} & \text{if } \mathsf{G} = \mathsf{i}\colon (\mathsf{G}_1 | \mathsf{G}_2),\ \mathsf{G}_1\!\downarrow_\mathsf{A}^{\overline{q_0},\overline{q_e}} \cap\, \mathsf{G}_2\!\downarrow_\mathsf{A}^{\underline{q_0},\underline{q_e}} = \emptyset,\ q_0 = (\overline{q_0}, \underline{q_0}) \text{ and } q_e = (\overline{q_e}, \underline{q_e}) \end{cases}$$

The following theorem shows that the system made of the projections of a g-choreography G is deadlock free if $[\![\mathsf{G}]\!]$ is defined.



(a) $\mathsf{A} \xrightarrow{\mathsf{x}} \mathsf{B}$ (b) $\mathsf{A} \xrightarrow{\mathsf{y}} \mathsf{B}$ (c) $\mathsf{A} \xrightarrow{\mathsf{x}} \mathsf{B} | \mathsf{A} \xrightarrow{\mathsf{y}} B$

Figure 5: Examples of projections

Theorem 1. *For a* $\mathsf{G} \in \mathscr{G}$ *let* s_0 *be the initial state of the communicating system* $(min(\mathsf{G}\downarrow_{\mathsf{A}}^{q_{0\mathsf{A}},q_{e\mathsf{A}}}))_{\mathsf{A}\in\mathsf{P}}$. *If* $[\![\mathsf{G}]\!] \neq \bot$ *and* $s_0 \stackrel{e_0}{\Rightarrow} \ldots \stackrel{e_{n-1}}{\Longrightarrow} s_n$ *then* s_n *is not a deadlock.*

Proof sketch. The proof of the theorem is done by structural induction over the syntax of g-choreography. The base cases are straightforward, since the projection of a empty choreography or of a single interaction can not lead to a deadlock. For the inductive steps, we rely on the fact that minimization of CMFS preserves the language of the communicating system and does not introduce deadlocks. For sequential and parallel composition, the proof is done by showing that if there is a deadlock in the composed communicating system, then there must be a deadlock in at least one of the constituent systems. This holds straigthforwardly for the sequential composition. For the parallel composition, we note that

- in each thread, every output of a message, say m, has a corresponding input action in a receiving machine, say A;
- the machine M_{A} of the receiver A is the product of the threads on A.

Therefore, the configurations where the message m is sent have to reach a configuration where A has the reception of m enabled (otherwise in one of the threads there would be a deadlock). Hence, eventually m will be consumed.

For the non-deterministic composition, we show that if there is a trace in system S made of machines $(\mathsf{G}_1 + \mathsf{G}_2)$ with $\mathsf{A} \in \mathsf{P}$, then there must be the same trace in one of the systems made of machines $\mathsf{G}_1\downarrow_{\mathsf{A}}^{q_0,q_e}$ or $\mathsf{G}_2\downarrow_{\mathsf{A}}^{q_0,q_e}$. This is due to the well-branched condition. If participant B selects $\mathsf{G}_i\downarrow_{\mathsf{A}}^{q_0,q_e}$ in the communicating system S then all other participants are forced to follow the same choice. This allows us to build a simulation relation between the communicating system of the non-deterministic choice and the one consisting of the CMFS $(\mathsf{G}_i\downarrow_{\mathsf{A}}^{q_0,q_e})_{\mathsf{A}\in\mathsf{P}}$. □

The following theorem shows that the traces of the system made of the projections of a g-choreography G are included in the language of the g-choreography if $[\![\mathsf{G}]\!]$ is defined.

Theorem 2. *For a* $\mathsf{G} \in \mathscr{G}$ *let* $S = (min(\mathsf{G}\downarrow_{\mathsf{A}}^{q_{0\mathsf{A}},q_{e\mathsf{A}}}))_{\mathsf{A}\in\mathsf{P}}$. *If* $[\![\mathsf{G}]\!] \neq \bot$ *then* $\mathscr{L}[S] \subseteq \mathscr{L}[\mathsf{G}]$.

Proof sketch. The theorem is proved by structural induction over the syntax The theorem is proved by structural induction over the syntax of the g-choreographies. The two main tasks are to show that (*i*) the dependencies are preserved in the case of sequential composition and (*ii*) no additional communication occurs in the case of parallel composition. For the sequential composition we proceed as follows. By definition, every word w_0 in $\mathscr{L}[\mathsf{G};\mathsf{G}']$ is the shuffling of two words, $w \in \mathscr{L}[\mathsf{G}]$ and $w' \in \mathscr{L}[\mathsf{G}']$. Additionally, the side condition of the semantics of sequential composition ensures that all the events of w having subject A precede in w_0 every event of w' with subject A. For the second task we rely on the fact $[\![\mathsf{G}]\!]$ is defined and we follow the same reasoning done for Theorem 1. □

In general, the converse of the inclusion in Theorem 2, that is $\mathscr{L}[\mathsf{G}] \subseteq \mathscr{L}[S]$, does not hold. The reason is due to the fact that semantics of parallel composition of g-choreographies does not assume a FIFO policy on channels. In fact, the communicating system can have less behaviors than the interleaving of the two constituent threads because of the additional dependencies imposed by FIFO channels. For instance, take the g-choreography $\mathsf{G} = \mathsf{A} \xrightarrow{\mathsf{x}} \mathsf{B} | \mathsf{A} \xrightarrow{\mathsf{y}} \mathsf{B}$; the word $\mathsf{AB!xAB!yAB?yAB?x}$ is in $\mathscr{L}[\mathsf{G}]$ but it is not in $\mathscr{L}[(min(\mathsf{G} \downarrow_{\mathsf{A}}^{q_{0\mathsf{A}}, q_{e\mathsf{A}}}))_{\mathsf{A} \in \mathsf{P}}]$.

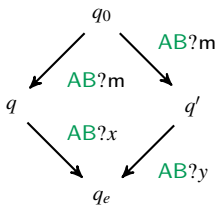
8 Conclusions

We introduced an abstract semantics of choreographies expressed as global graphs. We defined the new semantics without makig assumptions on how messages are exchanged at lower levels. We showed that our abstract semantics is adequate by demonstrating how projections preserve it on communicating machines. A distinguishing feature of this proposal is that it fixes a specification language of global artefacts that is not a dependent variable of the semantics of the local views.

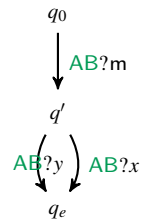
Our framework seems to be more expressive than existing ones; it allows the same participant to operate in both threads of the parallel composition and it does not force passive participants to receive a message signalling the selected choice as first operation in a non-deterministic composition. This is possible due to the well-branched condition. Interestingly, this condition is parametric and depends on the strategy used to find the bijection required by reflection. This can range from always straightforwardly using the empty bijection (thus enforcing the same syntactical constrains of the existing proposals) to finding a graph isomorphism. A projection algorithm, different from the one proposed here, can reuse the mechanism used to check the well-branched condition to identify the common behavior of participants and avoid using minimization.

The independence of the global semantics from the local one is evident from Theorem 2. We regard as a good property of our semantics the fact that global artefacts have "more executions" than the local ones obtained from their projections. Intuitively, this amounts to say that projections are refinements of (more abstract) global view. Another advantage of this is that changing local artifacts (e.g. using a CMFS where buffers are not FIFO, in which case the other inclusion of Theorem 2 would also hold) does not require to modify the semantics of the global view.

An interesting future direction is to explore alternative projection algorithms. We plan to define projections that exploit reflections. This could be better explained by observing what happens when projecting the simple choregraphy $\mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}; \mathsf{A} \xrightarrow{\mathsf{x}} \mathsf{B} + \mathsf{A} \xrightarrow{\mathsf{m}} \mathsf{B}; \mathsf{A} \xrightarrow{\mathsf{y}} \mathsf{B}$, say on participant B (we ignore control points because immaterial). Our algorithm yields the following machine:



which after minimisation becomes



However, exploiting the bijection of the reflection, one could directly obtain the machine on the right (avoiding the cost of minimising machines). Note that other projection algorithms capable of handling of handling the example above (as e.g., the one in [12]) also require minimisation, while projections based on types (as e.g., the ones in [10]) are undefined on the previous example because they require prefixes of branches to be pairwise different.

To simplify the presentation we used loop-free global graphs. However, all results presented here can be easily extended to graphs with structured loops that are represented as repetitions of g-choreography. This is possible since the semantics side-conditions do not depend on the (possibly infinite) language of the choreography, but rather on the hypergraps, which is finite.

References

[1] Charlton Barreto & et al. (2007): *Web Services Business Process Execution Language Version 2.0*. https://www.oasis-open.org/committees/download.php/23964/wsbpel-v2.0-primer.htm.

[2] Davide Basile, Pierpaolo Degano, Gian-Luigi Ferrari & Emilio Tuosto (2016): *Relating two automata-based models of orchestration and choreography*. *JLAMP* 85(3), pp. 425 – 446.

[3] Laura Bocchi, Hernán C. Melgratti & Emilio Tuosto (2014): *Resolving Non-determinism in Choreographies*. In: *ESOP*, pp. 493–512.

[4] Daniel Brand & Pitro Zafiropulo (1983): *On Communicating Finite-State Machines*. *JACM* 30(2), pp. 323–342. Available at http://doi.acm.org/10.1145/322374.322380.

[5] Marco Carbone, Kohei Honda & Nobuko Yoshida (2007): *A Calculus of Global Interaction based on Session Types*. *Electronic Notes in Theoretical Computer Science* 171(3), pp. 127 – 151.

[6] Giuseppe Castagna, Mariangiola Dezani-Ciancaglini & Luca Padovani (2012): *On Global Types and Multi-Party Session*. *LMCS* 8(1).

[7] Mario Coppo, Mariangiola Dezani-Ciancaglini, Nobuko Yoshida & Luca Padovani (2016): *Global progress for dynamically interleaved multiparty sessions*. *Mathematical Structures in Computer Science* 26(2), pp. 238–302.

[8] Mila Dalla Preda, Maurizio Gabbrielli, Saverio Giallorenzo, Ivan Lanese & Mauro Jacopo (2015): *Dynamic Choreographies - Safe Runtime Updates of Distributed Applications*. In: *COORDINATION 2015*, pp. 67–82.

[9] Pierre-Malo Deniélou & Nobuko Yoshida (2012): *Multiparty Session Types Meet Communicating Automata*. In: *ESOP*, pp. 194–213.

[10] Kohei Honda, Nobuko Yoshida & Marco Carbone (2016): *Multiparty Asynchronous Session Types*. *J. ACM* 63(1), pp. 9:1–9:67.

[11] Nickolas Kavantzas, Davide Burdett, Gregory Ritzinger, Tony Fletcher & Yves Lafon (2004): *Web Services Choreography Description Language Version 1.0*. http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217.

[12] Julien Lange, Emilio Tuosto & Nobuko Yoshida (2015): *From Communicating Machines to Graphical Choreographies*. In: *POPL15*, pp. 221–232.