Walking Through the Semantics of Exclusive and Event-Based Gateways in BPMN Choreographies

Flavio Corradini, Andrea Morichetta, Barbara Re, and Francesco Tiezzi

School of Science and Technology, University of Camerino, Camerino, Italy
{*name.surname*}@unicam.it

Abstract. With the evolution of distributed systems, nowadays BPMN choreography diagrams have acquired more and more importance for modelling systems interaction. However, one of the drawbacks of this model is the lack of formal semantics, which leads to different interpretations, and hence implementations, of some of its features. Among the BPMN choreography elements, particularly ambiguous is the semantics of the exclusive and event-based gateways, used to represent different forms of choices. Formalisations of these elements have been proposed in the literature, but none of them is derived from a direct and faithful modelling of the description provided by the BPMN standard. In this work, instead, we provide a direct formalisation, in terms of an operational semantics, that aims at shedding light on the intricacies of the behaviour of the exclusive and event-based gateways. The effectiveness of the approach is shown by illustrating how our semantics can disambiguate tricky behaviours in choreography models.

Keywords: BPMN 2.0, Choreographies, Exclusive and Event-Based Gateways, Operational Semantics.

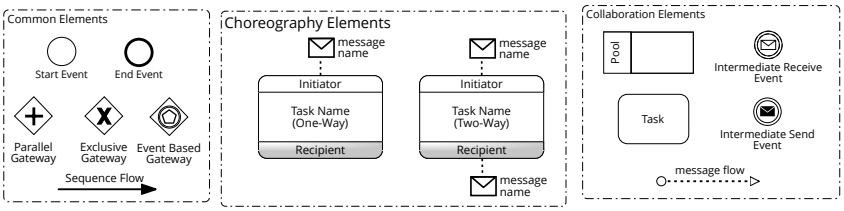
1 Introduction

The BPMN 2.0 OMG standard [23] (in the following just BPMN) is more and more adopted by academia and industry as a modelling language for distributed systems. Its diffusion is mainly due to its capability to describe the behaviour of components by means of an appealing graphical notation. In particular, a BPMN *choreography* diagram provides a global specification focusing on component interactions, while a BPMN *collaboration* diagram describes the implementation of every single component, possibly deployed and managed by different organisations, in terms of exchanged messages and internal behaviour. In such a setting, organisations that are willing to cooperate to achieve a specific objective can refer to choreography specifications for describing the interactions between different parties. On the other hand, interested organisations can put in place the cooperation by relying on collaboration models able to describe both the communication patterns and the internal activities of a component.

The community widely accepts the BPMN standard for its expressiveness and adaptability to many fields. Despite this advantage, its complex semantics can generate possible flaws during the design phase [4] [7]. This phenomenon is accentuated by the lack of rigour of the standard in the element descriptions, due to the use of natural language for the semantic definitions. This problem is witnessed by the differences that can be observed between the available business process management systems: from one implementation to another different semantics are used for the same element, disorienting the expectation of the designer [11]. This lack of transparency in the standard, joint with the inability of being executable, has reduced the adoption of BPMN choreography diagrams in favour of collaboration diagrams that played for years a more relevant role [2]. Despite this, BPMN choreography is considered the most appropriate notation to model the coordination of the interactions between different participants when the process cannot be controlled in a centralised manner [3]. Moreover, the increasing diffusion of blockchain technology, natively implementing decentralised trusted scenarios, is asking for adequate modelling languages [13]. Choreographies are promising modelling abstractions for contracts between two or more organisations, being both communication-centric and human understandable by business analysts and IT specialists. The whole area of choreographies may be revitalised by such technological evolution [2]. This raises the need of making even clearer the semantics of choreography diagrams before their wider adoption.

In a distributed scenario, without a clear semantics for choreographies, the difficulty on producing high-quality models for the global perspective may represent a barrier in the development of each single components. The problem is further compounded by the fact that ambiguities on the semantics also regard largely used elements, such as the *exclusive* and the *event-based gateways*, exploited to express in BPMN choreographies different forms of choice among alternative execution paths. To fill this gap, in this paper we first present a detailed analysis of the the natural language descriptions provided by the BPMN standard about the semantics of key choreography elements. We specifically focus on the two gateway elements mentioned above. We then propose an informal characterisation, by resorting to a graphical notation, of the semantics of the two gateways, which combines all requirements stated in the standard. Finally, to provide a clear understanding of the ambiguous points of the standard, we provide a formalisation of the BPMN choreographies semantics, given in terms of an operational semantics defined on top of a textual representation of the models. The proposed work is motivated and validated by relying on an example coming from the literature and presenting an evident issue caused by a misunderstanding of the standard.

The rest of the paper is organised as follows. Section 2 provides background notions on BPMN choreographies and collaborations. Section 3 motivates our work by detailing issues resulting from the standard. Section 4 informally describes the semantics of exclusive and event-based gateways. Section 5 proposes the formalisation of choreography syntax and semantics. Section 6 discusses related works. Finally, Section 7 concludes the paper and discusses future work.



a. Flow elements b. Choreography tasks c. Collaboration elements

Fig. 1. BPMN 2.0 Elements.

2 Background

This section presents the relevant elements of choreography and collaboration diagrams we use in the paper.

Figure 1(a) depicts the most used modelling elements that can be included in both diagrams. **Events** are used to represent something that can happen. An event can be a *start event*, representing the point in which the choreography/collaboration starts, while an *end event* is raised when the choreography/collaboration terminates. **Gateways** are used to manage the flow of a choreography/collaboration both for parallel activities and choices. Gateways act as either join nodes (merging incoming sequence edges) or split nodes (forking into outgoing sequence edges). Different types of gateways are available. A *parallel gateway (AND)* in join mode has to wait to be reached by all its incoming edges to start, and respectively all the outgoing edges are started simultaneously in the split case. An *exclusive gateway (XOR)* describes choices; it is activated each time the gateway is reached in join mode and, in split mode, it activates exactly one outgoing edge. An *event-based gateway* is similar to the XOR-split gateway, but its outgoing branches activation depends on the occurrence of a catching event in the collaboration and on the reception of a message in the choreography; these events/messages are in a race condition, where the first one that is triggered wins and disables the other ones. **Sequence Flows** are used to connect collaboration/choreography elements to specify the execution flow.

Focusing on the choreography diagram, we underline its ability to specify the message exchanges between two or more participants. This is done by means of **Choreography Tasks** in Figure 1(b). They are drawn as rectangles divided in three bands: the central one refers to the name of the task, while the others refer to the involved participants (the white one is the initiator, while the gray one is the recipient). Messages can be sent either by one participant (One-Way tasks) or by both participants (Two-Way tasks). Concerning choreography tasks, we rely on the follow design choices. In relation to the *Two-Way choreography task*, the OMG standard states that it is "an atomic activity in a choreography process" execution [23, p. 323]. However, this does not mean that the task blocks the whole execution of the choreography. In fact, participants are usually distributed, and we assume that other choreography tasks involved in different parallel paths of the choreography can be executed. Thus, here we intend atomicity to mean that both messages exchanged in a Two-Way task have to be

received before triggering the execution along the sequence flow outgoing from the task. Therefore, even if we allow Two-Way tasks in the choreography models, we safely manage them as pairs of One-Way tasks preserving the same meaning.

In a collaboration diagram, together with the flow elements, the elements in Figure 1(c) can be included. **Pools** are used to represent participants involved in the collaboration. **Tasks** are used to represent specific works to perform within a collaboration by a participant. **Intermediate Events** represent something that happens during the flow of the process, such as sending or receiving of a message. **Message Edges** are used to visualize communication flows between participants, by connecting communication elements within different pools.

3 Motivations

A choreography model represents a guideline for driving the communication interactions between organisations and represents a reference point for the implementation of each single component. For this reason a shared and clear understanding of the meaning of BPMN elements is needed in order to improve the quality of the designed model.

Unfortunately, such a common understanding cannot be taken for granted. Even if we deal with BPMN, which being a standard language should guarantee a certain level of rigorousness, the practice gives evidences of various issues. Looking in the literature, only few choreography models are available. Most of them are partial specifications, which miss to specify, e.g., messages and conditions. Moreover, when these are included, they are often incorrectly used, due to misunderstandings resulting by inaccuracies and inconsistencies in the standard. A typical example of such problems concerns the exclusive and event-based gateways, two elements that despite their tricky semantics are largely used in choreography diagrams. As a reference example, we report in Figure 2 an erroneous choreography model drawn from the literature [24]. It shows the interactions between two participants: the *SugarPerson*, asking for sugar, and *SugarGrid*, looking for sugar. The model under consideration contains a mistake on the

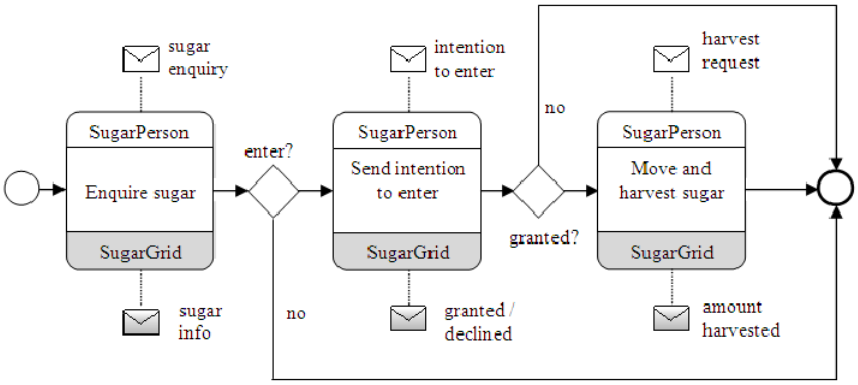


Fig. 2. An erroneous choreography model (source [24]).

- The data used for **Gateway** *Conditions* MUST have been in a **Message** that was sent prior to (upstream from) the **Gateway**.
  - More specifically, all *Participants* that are directly affected by the **Gateway** MUST have either sent or received the **Message(s)** that contained the data used in the *Conditions*.
    - Furthermore, all these *Participants* MUST have the same understanding of the data. That is, the actual values of the data cannot selectively change after a *Participant* has seen a **Message**. Changes to data during the course of the **Choreography** MUST be visible to all the *Participants* affected by the **Gateway**.

Fig. 3. Exclusive gateway [23, p. 345].

specification of the first exclusive gateway. In particular, the enforcement of the *"enter?"* condition is not correct. In fact, the information about the intention of the *SugarPerson* to enter in the sugar grid is sent by the *SugarPerson* to the *SugarGrid* in the *"Send intention to enter"* task, which is executed after the execution of the gateway. The exclusive gateway, as explained in the next section, requires all participants involved in the subsequent tasks to be able to take the same decision expressed by the gateway condition. But, in this example, the decision cannot be properly taken by the *SugarGrid* participant, since he does not have yet the information about the *SugarPerson*'s intention. This model indeed violates a prescription of the standard [23, p. 345], requesting that the value used by the gateway in the condition evaluation has to be included in a message sent before the gateway execution.

Let us now focus more deeply on the description of choreography diagrams provided by the BPMN standard. We comment on observed contradictions that can be misleading for designers intending to use it. Here in the following, we report some excerpt focusing on the use of exclusive and event-based gateways.

Figure 3 is an extract of the standard [23, p. 348] describing the general behaviour of the exclusive gateway. The three bullet points stress the importance to share control information between all participants before to be used in decisions. The text is supported in the standard by two examples representing a choreography interaction between three participants and the corresponding implementation through collaboration, reported in Figure 4 and 5, respectively. Few pages later, the standard [23, p. 348] also states that a choreography configuration can be valid only if all the participants have a shared information upon which the decision is made. Figure 6 shows an excerpt. At this point a first contradiction in the standard has been detected. It is between the text in Figure 3 and Figure 6, and the example in Figure 5. The text stresses the need to share knowledge through messages between senders and receivers, while in Figure 5 the participant C does not receive any message by participant A and only the message *M1* is exchanged between participants A and B.

Reading more about the description in Figure 7 coming from [23, p. 349] we observe some constraints on the business process implementing a choreography. If we check the alignment between the text and the model in Figure 5 we observe some issues. In particular, the text confirms the usage of the exclusive gateway in the process of the participant that is the initiator but erroneously state about the usage of the event-based gateway for the receiver participants (see point 3). The text in Figure 7 is also misaligned with respect to the general description of exclusive gateway in Figure 3.

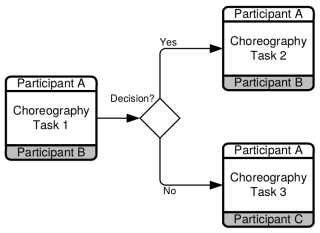


Fig. 4. An example of the exclusive gateway [23, p. 348].

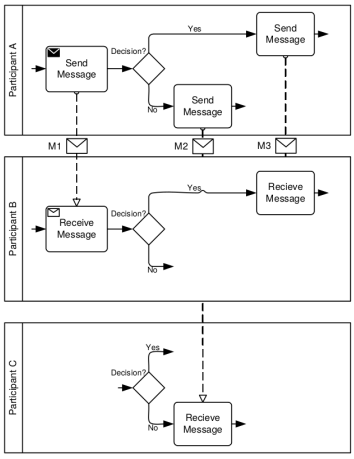


Fig. 5. The corresponding collaboration view of the choreography in Fig. 4 [23, p. 349].

This configuration can only be valid if all the *Participants* in the **Choreography Activities** that follow the **Gateway** have seen the data upon which the decision is made. If either "Participant B" or "Participant C" had not sent or received a **Message** with the appropriate data, then that *Participant* would not be able to know if they are suppose to receive a **Message** at that point in the **Choreography**. There is also the assumption that the value of the data is consistent from the point of view of all *Participants*.

Fig. 6. Valid choreography configuration [23, p. 348].

The REQUIRED execution behavior of the **Gateway** and associated **Choreography Activities** are enforced through the **Business Processes** of the *Participants* as follows:

- Each **Choreography Activity** and the **Sequence Flow** connections are reflected in each *Participant* **Process**.
- The **Gateway** is reflected in the **Process** of each *Participant* **Process** that is an initiator of **Choreography Activities** that follow the **Gateway**.
- For the receivers in **Choreography Activities** that follow the **Gateway**, an **Event-Based Gateway** is used to consume the associated **Message** (sent as an outcome of the **Gateway**). When a *Participant* is the receiver of more than one of the alternative **Messages**, the corresponding receives follow the **Event-Based Gateway**. If the *Participant* is the receiver of only one such **Message**, that is also consumed through a receive following the **Event-Based Gateway**. This is because the *Participant* **Process** does not know whether it will receive a **Message** (since the **Gateway** entails a choice of outcomes).

Fig. 7. Text Describing Collaboration in Fig. 5 [23, p. 349].

Another confirmation of the inaccuracy of the text in Figure 7 can be obtained by reasoning on the definition of the event-based gateway reported in Figure 8. It seems that there is no difference in the implementation of the exclusive gateway and event-based gateway when passing from a choreography to a collaboration model. This is obviously not possible, and the recognised overlap is a clear contradiction of the standard.

Summing up, in some parts the standard seems to be written with few attention. Moreover, the used examples are not complete and in most of the cases are inaccurate to substantiate the text. Finally, we believe that a major stumbling block is the language in which the standard is defined: natural language is inadequate when used to define the semantics of choreography elements.

The REQUIRED execution behavior of the **Event-Based Gateway** and associated **Choreography Activities** are enforced through the **Business Processes** of the *Participants* as follows:

- Each **Choreography Activity** and the **Sequence Flow** connections is reflected in each *Participant* **Process**.
- If the senders following the **Gateway** are the same, the **Event-Based Gateway** is reflected as an **Exclusive Gateway** in that *Participant's* **Process**. This is because the choice of which **Message** to send is determined by the same Participant. If the senders are different, sending occurs through different **Processes**.
- If the receivers are the same, the senders can be the same or different. In this case, the **Event-Based Gateway** is reflected in the receiver's **Process**, with the different **Message** receives following the **Gateway**.
- If the receivers are different, the senders need to be the same. The **Event-Based Gateway** is reflected for different receiver **Processes** such that the respective receive follows the **Gateway**. A time-out can be used to ensure that the **Gateway** does not wait indefinitely.

Fig. 8. Event-Based Description [23, p. 351]

4 Discussion on Exclusive and Event-Based Gateways

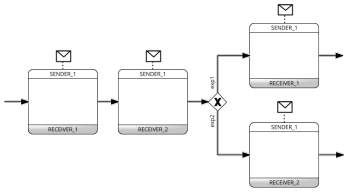
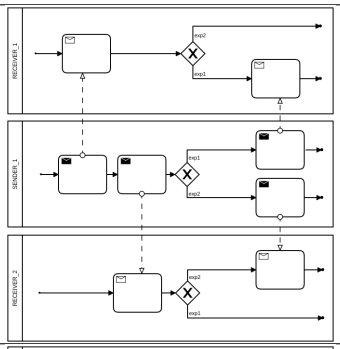
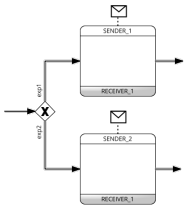
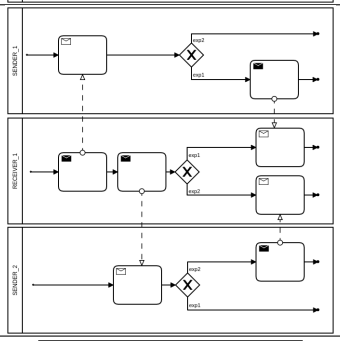
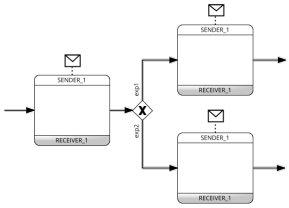
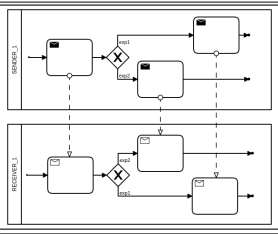
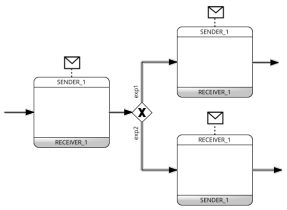
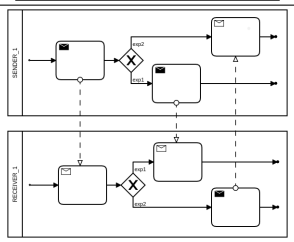
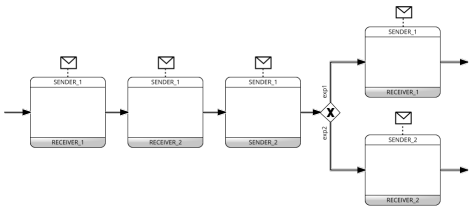
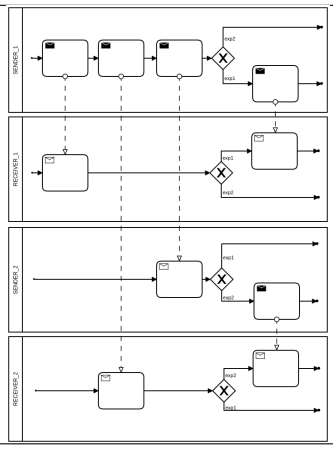
In this section, we informally clarify the semantics of the choreography diagram discussing the correct implementation of the exclusive gateway and event-based gateway through examples mapping choreographies to collaborations.

To make clear the principle at the basis of the choreography design, we first discuss the *data management* and *realizability*. Data in choreography does not have any control mechanism, so they are not maintained in any central source. The only way to share information is the exchange of messages. On the other hand, to be realizable [5] the choreography should respect a general rule where the initiator of a choreography activity must have been involved as initiator or receiver in the previous choreography activity. This restriction limits the combination of participants in the choreography task and in particular when the tasks sequence includes a gateway in the middle.

Focusing on the admitted combinations of participants, moving from the left-hand-side of a gateway to the right-hand side, we can have five different possible configurations: (i) same sender and same receiver; (ii) same sender and different receivers (iii) different senders and same receiver; (iv) same sender and same receiver swapped; (v) different senders and different receivers.

Tables 1 shows on the left column of each sub-table the possible choreography configurations of the exclusive gateway, and on the right column the respective collaborations implementation. In particular, Table 1(a) depicts the design of a choreography containing an exclusive gateway in the configuration of "same senders and different receivers". The corresponding collaboration should contain a first sharing of information from the sender S1 to the receivers R1 and R2, and successively all participants will perform the same choice using the exclusive gateways with the same condition. Table 1(b) represent the configuration with different senders and same receiver. In this case, the task before the gateway was omitted just for the sake of simplicity, but we have always to respect the general rule defining the continuity between participants involved in consecutive tasks. In this configuration, both senders will deliver a message towards the receiver, and it will select the desired message ignoring the others, admitting message loss. The third configuration in Table 1(c) is with same sender and same receiver. The messages exchanged before and after the gateway involve the same participants.

Table 1. Exclusive Gateway Implementation

	Choreographies	Collaborations
a)		
b)		
c)		
d)		
e)		

$$
\begin{aligned}
C ::= {} & \mathsf{start}(\mathsf{id},\mathsf{e}) \mid \mathsf{end}(\mathsf{id},\mathsf{e}) \mid \mathsf{andSplit}(\mathsf{id},\mathsf{e},E) \mid \mathsf{andJoin}(\mathsf{id},E,\mathsf{e}) \\
& \mid \mathsf{xorSplit}(\mathsf{id},\mathsf{e},(\mathsf{e}_1,\mathsf{exp}_1),\ldots,(\mathsf{e}_k,\mathsf{exp}_k)) \mid \mathsf{xorJoin}(\mathsf{id},E,\mathsf{e}) \mid \mathsf{task}(\mathsf{id},\mathsf{e}_1,\mathsf{e}_2,\mathsf{p},\mathsf{p}',\mathsf{m}) \\
& \mid \mathsf{eventBased}(\mathsf{id},\mathsf{e},(\mathsf{e}_1,\mathsf{p}_1,\mathsf{p}'_1,\mathsf{m}_1),\ldots,(\mathsf{e}_k,\mathsf{p}_k,\mathsf{p}'_k,\mathsf{m}_k)) \mid C_1|C_2
\end{aligned}
$$

Fig. 9. Syntax of BPMN Choreography Structures.

The sender will communicate to the receiver its will for the choice, and in the implementation both parties will follow the same path according to the shared information.

The configuration in Table 1(d) is similar to the previous one except for the sender and receiver that are swapped on the right-hand side of the gateway. Again, once the information at the basis of the decision is communicated, the two participants can continue according to either sending a message or waiting for another one. The last case in Table 1(e) represents the configuration with different senders and different receivers. Of course, this can be possible only respecting the right sequence of messages between the involved participants. Despite two separated communications, the participants will move according to the shared information in a coordinated way. Notably, these last two cases are not always possible, since we should consider the structure of the sequence task.

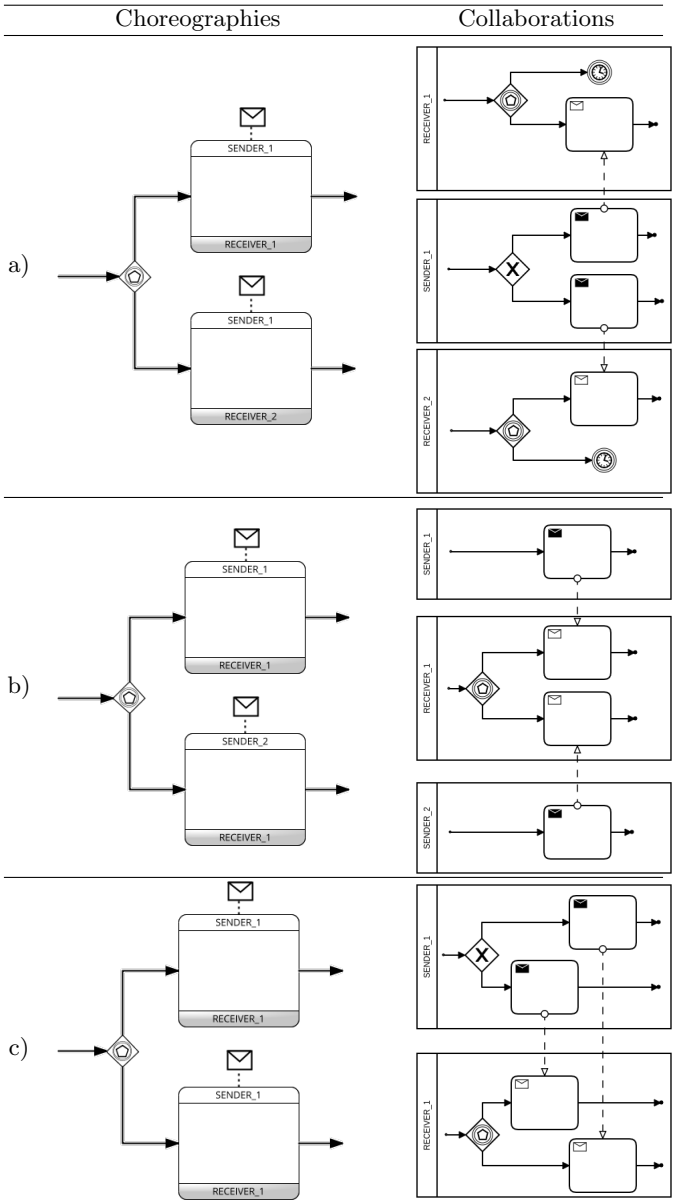
Table 2 shows on the left column the three choreography configurations that are admitted for the event-based gateway, and on the right column the respective collaboration implementations. Differently from the exclusive gateway, here it is not required the sharing of information before a choice. In Table 2(a) it is depicted the case with same sender and different receivers. The sender is the unique participant that takes the decision, and the receivers react consequently. In the implementation, a timer is used for avoiding the deadlock of the receiver not considered. Table 2(b) depicts the configuration with different senders and same receiver. Here the event-based gateway is used in the receiver implementation, and the logic of senders is simply the delivery of the message. Here we are in front of a race condition, where the late message is lost. The configuration with same sender and same receiver depicted in Table 2(c) is the standard communication where the sender takes the decision using an exclusive gateway and the receiver acts according to the choice taken by the counterpart.

5 Formal Semantics

This section presents our formalisation of BPMN choreographies, which in particular deals with the peculiarities of the event-based and exclusive gateways. The aim is to shed light on the semantics of these elements, in a way as much faithful as possible with the informal semantics provided by the BPMN standard discussed in the previous section. To enable a formal treatment, we defined a Backus Normal Form (BNF) syntax providing a textual representation of the structure of BPMN choreographies, on top of which we have defined the operational semantics of the BPMN choreography language.

The BNF syntax of the choreography models structure is given in Figure 9. In the grammar, the non-terminal symbol C represents *Choreography Structures*,

Table 2. Event-Based Gateway Implementation



while the terminal symbols, denoted by the sans serif font, are the considered elements of a BPMN model, i.e. events, gateways and tasks. Notably, we are not proposing a new modeling formalism, but we are only using a textual notation for the BPMN elements. With respect to the graphical notation, the textual one is more manageable for supporting the formal definition of the semantics.

As a matter of notation, $\mathsf{e} \in \mathbb{E}$ denotes a *sequence edge*, $E \in 2^{\mathbb{E}}$ a set of edges, and $\mathsf{m} \in \mathbb{M}$ a message. We also use a set $\mathbb{EXP}$ of *expressions* (ranged over

by exp), whose precise syntax is deliberately not specified; we just assume that expressions contain *values* $\mathsf{v} \in \mathbb{V}$. Such design choice has been influenced by the fact that the expression language operating on data is left unspecified even by the BPMN standard. Moreover, id and p denote names uniquely identifying a model element and a participant, respectively.

The correspondence between the syntax used here and the graphical notation of BPMN is as follows.

- $\mathsf{start}(\mathsf{id}, \mathsf{e})$ represents a start event identified by id with outgoing edge e.
- $\mathsf{end}(\mathsf{id}, \mathsf{e})$ represents an end event identified by id with incoming edge e.
- $\mathsf{andSplit}(\mathsf{id}, \mathsf{e}, E)$ represents an AND split gateway identified by id with incoming edge e and outgoing edges E (with $|E| > 1$).
- $\mathsf{andJoin}(\mathsf{id}, E, \mathsf{e})$ represents an AND join gateway identified by id with incoming edges E (with $|E| > 1$) and outgoing edge e.
- $\mathsf{xorSplit}(\mathsf{id}, \mathsf{e}, (\mathsf{e}_1, \mathsf{exp}_1), \ldots, (\mathsf{e}_k, \mathsf{exp}_k))$ represents an XOR split gateway identified by id with incoming edge e and outgoing edges $\mathsf{e}_1, \ldots \mathsf{e}_k$ (with $k > 1$) associated to boolean expressions $\mathsf{exp}_1, \ldots, \mathsf{exp}_k$, each expression defining if the corresponding branch can be activated or not.
- $\mathsf{xorJoin}(\mathsf{id}, E, \mathsf{e})$ represents an XOR join gateway identified by id with incoming edges E (with $|E| > 1$) and outgoing edge e.
- $\mathsf{task}(\mathsf{id}, \mathsf{e}_1, \mathsf{e}_2, \mathsf{p}, \mathsf{p}', \mathsf{m})$ represents a one-way task identified by id with incoming edge e_1, outgoing edge e_2, and sending a message m from p_1 to p_2. Two-way tasks are rendered in our formal framework as pairs of one-way tasks, hence they are not explicitly included in the syntax.
- $\mathsf{eventBased}(\mathsf{id}, \mathsf{e}, (\mathsf{e}_1, \mathsf{p}_1, \mathsf{p}'_1, \mathsf{m}_1), \ldots, (\mathsf{e}_k, \mathsf{p}_k, \mathsf{p}'_k, \mathsf{m}_k))$ represents an event-based gateway identified by id with incoming edge e, and a list (with $k > 1$) of tasks to be processed. It is worth noticing that the definition of the task list is given by quadruples of the form $(\mathsf{e}, \mathsf{p}, \mathsf{p}', \mathsf{m})$, where e is the outgoing edge of the task, p and p' are interacting participants (sender and receiver, respectively) and m is the exchanged message.
- $C_1 \,|\, C_2$ represents the composition of elements, which permits to render a choreography structure in terms of a collection of elements.

To achieve a compositional definition, each sequence edge of the BPMN model is split in two parts: the part outgoing from the source element and the part incoming into the target element. The two parts are correlated by means of unique sequence edge names in the BPMN model. Notably, we only consider terms of the syntax that are derived from BPMN models.

The operational semantics we propose is given in terms of configurations of the form $\langle C, \sigma, \gamma \rangle$, where C is a choreography structure, $\sigma : \mathbb{E} \to \mathbb{N}$ is an *execution state function* mapping sequence edges to numbers of tokens, and $\gamma : \mathbb{M} \to \mathbb{V}$ is a *message state function* mapping messages to values. The execution state obtained by updating in the state σ the number of tokens of the edge e to n, written as $\sigma \cdot \{\mathsf{e} \mapsto \mathsf{n}\}$, is defined as follows: $(\sigma \cdot \{\mathsf{e} \mapsto \mathsf{n}\})(\mathsf{e}')$ returns n if $\mathsf{e}' = \mathsf{e}$, otherwise it returns $\sigma(\mathsf{e}')$. The message state obtained by updating in the state γ the value of a message m to a value v, written as $\gamma \cdot \{\mathsf{m} \mapsto \mathsf{v}\}$, is defined similarly. The *initial* states of a choreography, where all sequence edges are unmarked and

all message values are undefined, denoted respectively by σ_0 and γ_0, are formally defined as: $\sigma_0(\mathsf{e}) = 0 \quad \forall \mathsf{e} \in \mathbb{E}$, and $\gamma_0(\mathsf{m}) = \mathit{undef} \quad \forall \mathsf{m} \in \mathbb{M}$.

The operational semantics is defined by means of a labelled transition system (LTS) on choreography configurations, formalising the execution of a choreography in terms of marking evolution and message exchanges. The LTS is a triple $\langle \mathcal{C}, \mathcal{A}, \rightarrow \rangle$ where: $\mathcal{C}$ is the set of choreography configurations; $\mathcal{L}$, ranged over by l, is the set of *labels* (of transitions that choreography configurations can perform); and $\rightarrow \subseteq \mathcal{C} \times \mathcal{L} \times \mathcal{C}$ is the *transition relation*. As usual, we write $\langle C, \sigma, \gamma \rangle \xrightarrow{l} \langle C, \sigma', \gamma' \rangle$ to indicate that $(\langle C, \sigma, \gamma \rangle, l, \langle C, \sigma', \gamma' \rangle) \in \rightarrow$ and say that the choreography configuration $\langle C, \sigma, \gamma \rangle$ performs a transition labelled by l and becomes the configuration $\langle C, \sigma', \gamma' \rangle$. Since choreography execution only affects the current states, and not the choreography structure, for the sake of presentation, we omit the structure from the target configuration of the transition. Thus, a transition $\langle C, \sigma, \gamma \rangle \xrightarrow{l} \langle C, \sigma', \gamma' \rangle$ is written as $\langle C, \sigma, \gamma \rangle \xrightarrow{l} \langle \sigma', \gamma' \rangle$. A label l represents a computational step and is defined by the graphical representation of the executed BPMN element together with further information: the element id and, possibly, the set of participants involved in a decision and/or an exchange of message. Notably, despite the presence of labels, this has to be thought of as a reduction semantics, because labels are not used for synchronisation (as instead it usually happens in labelled semantics), but only for keeping track of the executed elements.

The transition relation over choreography configurations is defined by the rules in Figure 10. Before commenting on the rules, we introduce the auxiliary functions they exploit. Specifically, function $\mathit{inc} : \mathbb{S}_e \times \mathbb{E} \rightarrow \mathbb{S}_e$ (resp. $\mathit{dec} : \mathbb{S}_e \times \mathbb{E} \rightarrow \mathbb{S}_e$), where $\mathbb{S}_e$ is the set of execution states, allows updating a state by incrementing (resp. decrementing) by one the number of tokens marking an edge in the state. Formally, they are defined as follows: $\mathit{inc}(\sigma, \mathsf{e}) = \sigma \cdot \{\mathsf{e} \mapsto \sigma(\mathsf{e}) + 1\}$ and $\mathit{dec}(\sigma, \mathsf{e}) = \sigma \cdot \{\mathsf{e} \mapsto \sigma(\mathsf{e}) - 1\}$. These functions extend in a natural ways to sets of edges as follows: $\mathit{inc}(\sigma, \emptyset) = \sigma$ and $\mathit{inc}(\sigma, \{\mathsf{e}\} \cup E)) = \mathit{inc}(\mathit{inc}(\sigma, \mathsf{e}), E)$; the cases for *dec* are similar. The function $\mathit{eval} : \mathbb{EXP} \times \mathbb{S}_m \rightarrow \mathbb{V}$, where $\mathbb{S}_m$ is the set of message states, evaluates an expression with respect to a given message state; since the expression language is left unspecified, the definition of *eval* is unspecified as well. We also use the function $\mathit{update} : \mathbb{S}_m \times \mathbb{M} \rightarrow \mathbb{S}_m$ that updates a message state by assigning a value to a given message; formally, the function is defined as follows: $\mathit{update}(\gamma, \mathsf{m}) = \gamma \cdot \{\mathsf{m} \mapsto \mathsf{v}\}$ for $\mathsf{v} \in \mathbb{V}$. Finally, function $p : 2^{\mathbb{E}} \rightarrow \mathbb{P}$, where $\mathbb{P}$ is the set of participants, returns all participants involved in the first tasks encountered along the edges in the set passed as input parameter; this function can be simply defined on the syntax of the choreography structure of the model under consideration.

We now briefly comment on the operational rules. Rule *Start* starts the execution of a choreography when it is in its initial state. The effect of the rule is to increment the number of tokens in the edge outgoing from the start event. Rule *AndJoin* decrements the tokens in each incoming edge and increments the number of tokens of the outgoing edge, when each incoming edge has at least one token. Rule *XorSplit* is applied when a token is available in the incoming edge of

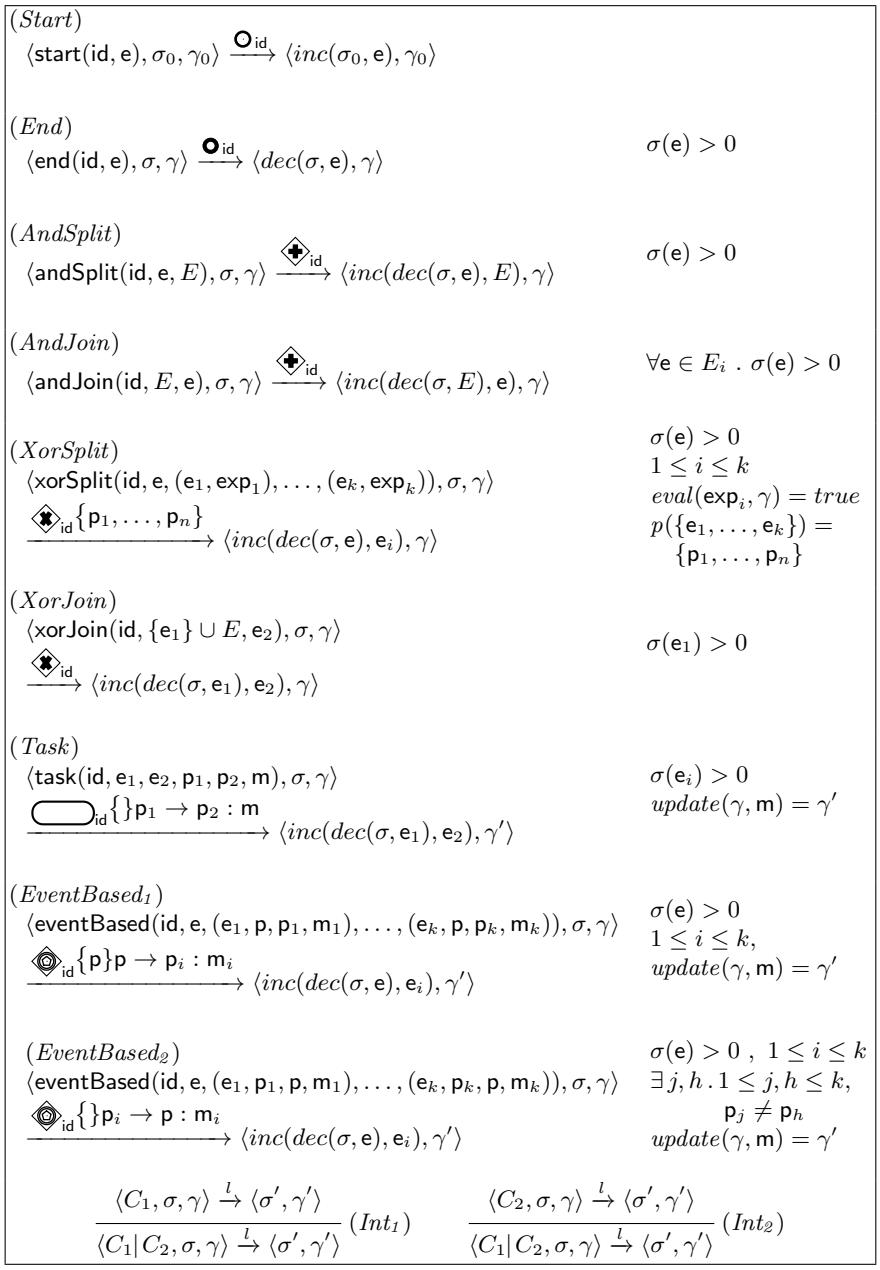


Fig. 10. Choreography Semantics.

an XOR split gateway; the rule decrements the token in the incoming edge and increments the tokens in one of the outgoing edges corresponding to a positive evaluation of the associated expression. The produced label reports the set of

all participants involved in the first tasks reachable from the gateway; indeed, in this case all such participants, with both sending and receiving roles, internally take the same decision in order to ensure the global behaviour prescribed by the choreography (see Table 1). Rule *XorJoin* is activated every time there is a token in one of the incoming edges, which is then moved to the outgoing edge.

Rule *Task* is activated when there is a token in the incoming edge of a choreography task, and moves the token from the incoming edge to the outgoing one. The rule produces a label describing the message exchange and indicating that no decision is taken by the participants involved in the communication. Moreover, the message state is updated with a new value for the involved message name; we abstract from the computation of the message value, which is indeed non-deterministically selected via the *update* function.

Notably, as prescribed by the BPMN standard [23, p. 315], the communication model is synchronous; indeed, according to the standard, a choreography task completes when the receiver participant reads the message. The rules for the event-based gateway are enabled each time there is a token in the incoming edge, which is moved to the activated outgoing edge. Moreover, the value of the exchanged message is updated in the message state. According to the involved participants we distinguish two cases: rule $EventBased_1$ is used in case the tasks following the gateway have the same sender (see Table 2(a) and Table 2(c)), while rule $EventBased_2$ is used in case the tasks have the same receiver and at least two distinguished senders (see Table 2(b)); notably, the case with the same sender and the same receiver is dealt with by the former rule.

Rule $EventBased_1$ produces a label recording, besides the message exchange, the fact that the sender participant has undertaken an internal decision, according to which the message to be sent has been selected. Instead, Rule $EventBased_2$ produces a label reporting just the information about the message exchange, as the decision about the message to be exchanged is not taken by a participant but it is the result of a race-condition. Finally, rules Int_1 and Int_2 deal with interleaving.

The proposed operational semantics has been conceived to clarify the behaviour of the exclusive and event-based gateways in BPMN choreographies. The formalisation of these elements is therefore intentionally articulate, since it aims at faithfully capturing their behaviour as informally described in the standard. Although the two gateways represent choice constructs, our operational rules significantly differ from the conditional and non-deterministic choice operators typically used in process algebras. Indeed, the semantics of the gateways depends on the form of interaction that can be carried out by the participants of the subsequent tasks, while this information does not affect the behaviour of the process algebraic constructs. Hence, the latter cannot be used to directly model the choices on BPMN choreographies. On the other hand, they provide a clearer semantics that facilitates the modelling activity to the choreography designers.

We conclude the section by showing our formalisation at work on the sugar harvesting example introduced in Section 3.

$\mathsf{start}(\mathsf{id}_1, \mathsf{e}_1)$
| $\mathsf{task}(\mathsf{id}_2, \mathsf{e}_1, \mathsf{e}'_1, \mathsf{SugarPerson}, \mathsf{SugarGrid}, \mathsf{sugarEnquiry})$
| $\mathsf{task}(\mathsf{id}'_2, \mathsf{e}'_1, \mathsf{e}_2, \mathsf{SugarGrid}, \mathsf{SugarPerson}, \mathsf{sugarInfo})$
| $\mathsf{xorSplit}(\mathsf{id}_3, \mathsf{e}_2, (\mathsf{e}_3, \mathsf{exp}_{\mathsf{yes}}), (\mathsf{e}_4, \mathsf{exp}_{\mathsf{no}}))$
| $\mathsf{task}(\mathsf{id}_4, \mathsf{e}_3, \mathsf{e}'_3, \mathsf{SugarPerson}, \mathsf{SugarGrid}, \mathsf{intentionToEnter})$
| $\mathsf{task}(\mathsf{id}'_4, \mathsf{e}'_3, \mathsf{e}_5, \mathsf{SugarGrid}, \mathsf{SugarPerson}, \mathsf{response})$
| $\mathsf{xorSplit}(\mathsf{id}_5, \mathsf{e}_5, (\mathsf{e}_6, \mathsf{granted}(\mathsf{response}) == \mathsf{yes}), (\mathsf{e}_7, \mathsf{granted}(\mathsf{response}) == \mathsf{no}))$
| $\mathsf{task}(\mathsf{id}_6, \mathsf{e}_6, \mathsf{e}'_6, \mathsf{SugarPerson}, \mathsf{SugarGrid}, \mathsf{harvestRequest})$
| $\mathsf{task}(\mathsf{id}'_6, \mathsf{e}'_6, \mathsf{e}_8, \mathsf{SugarGrid}, \mathsf{SugarPerson}, \mathsf{amountHarvested})$
| $\mathsf{xorJoin}(\mathsf{id}_7, \{\mathsf{e}_4, \mathsf{e}_7, \mathsf{e}_8\}, \mathsf{e}_9)$
| $\mathsf{end}(\mathsf{id}_8, \mathsf{e}_9)$

Fig. 11. Textual representation of the sugar harvesting choreography model.

Example 1. The BPMN choreography model in Figure 2 represents a scenario where a *SugarPerson* participant interacts with a *SugarGrid* participant in order to enter in a sugar grid and harvest the sugar in the grid. The model is rendered in our textual notation as shown in Figure 11 (we have specified unique element identifiers and edge names, which are omitted in the graphical representation).

In the graphical representation of the BPMN model, the decision of the first XOR gateway is abstracted by means of the *enter*? condition. According to the BPMN standard, XOR gateway decisions have to be based on the message data. In our case, focussing on the previously exchanged messages, we have to consider sugarEnquiry and/or sugarInfo. Let us consider only the latter message (the following reasoning does not change if we consider the former message or both of them). In our textual notation we would instantiate the expressions specified in the first XOR gateway (i.e., the element with identifier id_3) as follows:

$$\mathsf{exp}_{\mathsf{yes}} = (\mathsf{enter}(\mathsf{sugarInfo}) == \mathsf{yes}) \qquad \mathsf{exp}_{\mathsf{no}} = (\mathsf{enter}(\mathsf{sugarInfo}) == \mathsf{no})$$

where function enter() abstracts the decision to enter or not taken by the *SugarPerson*. According to our semantics, from the initial choreography configuration $\langle C, \sigma_0, \gamma_0 \rangle$, where C is the choreography structure in Figure 11, by applying rule *Start* and then twice rule *Task* (and by properly applying, each time, the interleaving rules), we can reach the configuration $\langle C, \sigma, \gamma \rangle$, with $\sigma = \sigma_0 \cdot \{\mathsf{e}_2 \mapsto 1\}$ and $\gamma = \gamma_0 \cdot \{\mathsf{sugarEnquiry} \mapsto \mathsf{v}_1\} \cdot \{\mathsf{sugarinfo} \mapsto \mathsf{v}_2\}$. Now, according to rule *XorSplit*, depending on the evaluation of the guard expressions, we can observe the following transitions:

$$\langle C, \sigma, \gamma \rangle \xrightarrow{\diamondsuit\!\!\!\!\times_{\mathsf{id}_3}\{\mathsf{SugarPerson}, \mathsf{SugarGrid}\}} \langle \sigma', \gamma \rangle$$

$$\langle C, \sigma, \gamma \rangle \xrightarrow{\diamondsuit\!\!\!\!\times_{\mathsf{id}_3}\{\mathsf{SugarPerson}, \mathsf{SugarGrid}\}} \langle \sigma'', \gamma \rangle$$

with $\sigma' = \sigma_0 \cdot \{\mathsf{e}_3 \mapsto 1\}$ and $\sigma'' = \sigma_0 \cdot \{\mathsf{e}_4 \mapsto 1\}$. Both transitions produce the same label, indicating that the entering decision is taken by both the *SugarPerson* and the *SugarGrid*. However, in this specific scenario, while the *sugarInfo* message data can allow the *SugarPerson* to take the decision, since this participant represents the person that actually decides whether to enter or not in the

grid, this data cannot be enough for the *SugarGrid* to decide whether the *SugarPerson* intends to enter or not. In fact, the intention to enter will be reported in the message *intentionToEnter*, which is sent by *SugarPerson* to the *SugarGrid* after the execution of the XOR gateway. Anyway, by using this message as argument of the expressions $\mathsf{exp}_{\mathsf{yes}}$ and $\mathsf{exp}_{\mathsf{no}}$, we will have that the configuration $\langle C, \sigma, \gamma \rangle$ will be deadlocked, because the evaluation of the two expressions will be undefined as $\gamma(\mathsf{intentionToEnter}) = \mathit{undef}$.

Our semantics has hence spotted a semantic inconsistency in the considered BPMN choreography model. Indeed, in this specific example, the designer has accidentally used the wrong element for the modelling the choice about entering in the sugar grid. The decision to use the event-based vs. the exclusive gateway in fact is strictly correlated to the availability of information for all involved participants, that in this case is absent for the *SugarGrid*. A possible fix to make the model compliant with the standard is to replace the XOR gateway with an event-based one. Since the *SugarGrid* has no knowledge about the intention of the *SugarPerson*, at this stage of the protocol it should be able to accept any decision coming from the counterpart. The suggested replacement falls in the specific case of Table 2(c), and is formally represented by the operational rule $\mathit{EventBased}_1$, which indeed produces a label indicating that the decision is taken only by the *SugarPerson*. □

6 Related Works

In this section, we first discuss observed issues of the BPMN language, then we discuss other choreography formalizations available in the literature.

BPMN 2.0 Observed Ambiguities. Due to its complexity, BPMN 2.0 standard has been studied to make more explicit the meaning of the elements and their semantics [8, 1, 9]. The investigation was also done through an analytic work highlighting the problems of the standard according to supported workflow patterns [7]. In particular, the author states that the standard is ambiguous due to the numerous underspecified descriptions of semantics for relevant concepts (e.g., data conditions, and data dependencies between processes). The highlighted problems are justified by the gap between conceptual and executable BPMN models and the fact that on average the designers use less than the 20% of the available elements. The same problem was discussed some years before in [22]. Other relevant studies investigate the standard and its support for systems implementation. In [16, 14, 17] the authors suggested a series of implementation clarifications useful to designers for a more accurate tool selection.

Choreography Formalisations. Formalisations of choreography syntax and semantics are proposed in different works [12, 18, 25, 15]. The authors of [12] present an efficient algorithm for extracting concrete choreographic programs with asynchronous messaging. They use the core choreography language, where the semantics is given in terms of labelled reductions. In [18] the authors propose a framework able to synthesize local code for processes starting from global choreographies containing the component ports, the model composition and the

coordination elements. In [25] the authors propose two abstract semantics of choreographies formalised as pomsets of communication events and as hypergraphs of events. In [15] the authors demonstrate that it is possible to perform reversible computation monitoring choreography models. Other formal semantics rely on types [19], programs [21], graphs [20, 6]. It is worth noticing that many proposed works provide the semantics by translation in other formalisms, while in our work we have preferred to develop a direct semantics. We believe indeed that extending available translations, like those into Petri Nets, may result in the generation of convoluted and large models, which may undermine the understanding of the formal meaning of the BPMN execution semantics, and their verification. In [10] we propose a deeper study based on direct formalizations for choreographies and collaborations, and considering their conformance via behavioural equivalences.

7 Concluding remarks

In this paper, we discuss in detail issues raised walking through the BPMN standard, with a specific focus on the exclusive and event-based gateways, which have a tricky semantics in case of choreography diagrams. These issues arose from the use in the standard of natural language for providing informal descriptions of the elements, and are emphasised by the lack of a clear semantics for choreography models. We discuss good practices in the use of the notation, and we provide a direct formalisation of the choreography elements in order to remove any ambiguity. As a future work, we intend to exploit the proposed formalisation to develop a modelling tool to support the designers during the specification of choreography models.

References

1. van der Aalst, W.M.: Business process management: a comprehensive survey. ISRN Software Engineering **2678**, 1–12 (2013)
2. et. al, J.M.: Blockchains for business process management - challenges and opportunities. ACM Trans. Management Inf. Syst. **9**(1), 4:1–4:16 (2018)
3. et. al, R.B.: Towards living inter-organizational processes. In: Business Informatics. pp. 363–366. IEEE Computer Society (2013)
4. Anna Suchenia et al.: Selected Approaches Towards Taxonomy of Business Process Anomalies, pp. 65–85. Springer (2017)
5. Basu, S., Bultan, T., Ouederni, M.: Deciding choreography realizability. In: POPL. pp. 191–202. ACM (2012)
6. Bertolino, A., Marchetti, E., Morichetta, A.: Adequate monitoring of service compositions. In: 9th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. pp. 59–69 (2013)
7. Börger, E.: Approaches to modeling business processes. Soft. & Syst. Modeling **11**(3), 305–318 (2012)

8. Chinosi, M., Trombetta, A.: BPMN: An introduction to the standard. Computer Standards & Interfaces **34**(1), 124–134 (2012)
9. Corradini, F., Fornari, F., Polini, A., Re, B., Tiezzi, F.: A formal approach to modeling and verification of business process collaborations. Sci. Comput. Program. **166**, 35–70 (2018)
10. Corradini, F., Morichetta, A., Polini, A., Re, B., Tiezzi, F.: Collaboration vs. choreography conformance in BPMN 2.0: From theory to practice. In: EDOC. pp. 95–104. IEEE (2018)
11. Corradini, F., Muzi, C., Re, B., Rossi, L., Tiezzi, F.: Global vs. Local Semantics of BPMN 2.0 OR-Join. In: SOFSEM. LNCS, vol. 10706, pp. 321–336. Springer (2018)
12. Cruz-Filipe, L., Larsen, K.S., Montesi, F.: The Paths to Choreography Extraction. In: FOSSACS. LNCS, vol. 10203, pp. 424–440. Springer (2017)
13. Dumas, M., Hull, R., Mendling, J., Weber, I.: Blockchain technology for collaborative information systems. Dagstuhl Reports **8**(8), 67–129 (2018)
14. Evéquoz, F., Sterren, C.: Waiting for the miracle: Comparative analysis of twelve business process management systems regarding the support of BPMN 2.0 palette and export. Tech. rep., HES-SO (2011)
15. Francalanza, A., Mezzina, C.A., Tuosto, E.: Reversible Choreographies via Monitoring in Erlang. In: DAIS. LNCS, vol. 10853, pp. 75–92. Springer (2018)
16. Geiger, M., Wirtz, G.: BPMN 2.0 serialization-standard compliance issues and evaluation of modeling tools. Enterprise Modelling and Information Systems Architectures (2013)
17. Gutschier, C., Hoch, R., Kaindl, H., Popp, R.: A pitfall with BPMN execution. In: WEB. pp. 7–13 (2014)
18. Hallal, R., Jaber, M., Abdallah, R.: From Global Choreography to Efficient Distributed Implementation. In: HPCS. pp. 756–763. IEEE (2018)
19. Honda, K., Yoshida, N., Carbone, M.: Multiparty asynchronous session types. J. ACM **63**(1), 9:1–9:67 (2016)
20. Lange, J., Tuosto, E., Yoshida, N.: From Communicating Machines to Graphical Choreographies. In: POPL. pp. 221–232. ACM (2015)
21. Mila Dalla Preda and Maurizio Gabbrielli and Saverio Giallorenzo and Ivan Lanese and Jacopo Mauro: Dynamic Choreographies - Safe Runtime Updates of Distributed Applications. In: COORDINATION. LNCS, vol. 9037, pp. 67–82. Springer (2015)
22. zur Muehlen, M., Recker, J.: How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In: Seminal Contributions to Information Systems Engineering, 25 Years of CAiSE, pp. 429–443. Springer (2013)
23. OMG: Business Process Model and Notation (BPMN V 2.0) (2011)
24. Onggo, B.S.: Agent-based simulation model representation using BPMN. In: Formal languages for computer simulation, pp. 378–400. IGI Global (2014)
25. Tuosto, E., Guanciale, R.: Semantics of global view of choreographies. JLAMP **95**, 17–40 (2018)