

# On the interplay between BPMN collaborations and the physical environment

**Abstract.** Nowadays, business scenarios are permeated and influenced by information from the surrounding physical environment that influences decision-making and vice-versa. This is especially evident in the presence of participants able to move in and interact with the physical environment. In this regard, BPMN is a widely accepted standard for representing multi-party business processes in terms of collaboration diagrams, offering an expressive and understandable notation. However, BPMN, as well as its extensions, overlook the support to the physical environment. In this paper, we describe the interplay between BPMN collaborations and the physical environment. Specifically, we bridge the BPMN meta-model with environmental concepts, defining environmental BPMN collaboration models. To facilitate a deeper understanding of the dynamics of these models, we provide a formal account of their semantics. We illustrate our findings through a fire-extinguishing collaborative scenario.

**Keywords:** BPMN · Physical environment · Meta-model · Formal semantics.

## 1 Introduction

Modern organizations continually seek efficient and effective ways to model, analyze, and optimize their business processes. In this regard, the Business Process Model and Notation (BPMN) [10] has gained widespread adoption as a standard to model in an intuitive and expressive way business processes. In particular, the so-called BPMN collaboration diagrams specify how different participants (e.g., humans, software systems, robots) behave, exchange information, and work together in the same environment to reach a common objective. This enhances clarity and transparency and enables the identification and management of inefficiencies or bottlenecks [6].

In modeling business processes it is crucial to consider the context where participants can operate [3]. According to [13], a context is defined as “the minimum of variables containing all relevant information that impact the design and execution of a business process”. In this work, we concentrate on the environmental context, which includes all information beyond the process control flow influencing the process [14]. In particular, we focus on the *physical* part of the environmental context, which is related to (i) the *space* where a process participant acts and moves, and (ii) the domain *attributes* that characterize it. For the sake of presentation, from now on we use the term *environment* to mean the union of space and attributes. Awareness of the environment plays an important role in determining the outcome of a business process [12]. Process participants are often part of the environment [11,15]: they can hold a position, move, react to a particular situation, or even make environmental changes. Therefore, activities, decisions, and events can depend on specific conditions of the environment

and influence it. More precisely, the status of the environment can (i) constrain the execution of an activity, e.g., a machinery starts only if the item to work on is in place; (ii) drive decision points, e.g., if a charge station is occupied, an autonomous forklift in a warehouse decides to wait in a safety zone; (iii) trigger events, e.g., an alarm triggers the termination of an activity or the process. At the same time, a process activity can influence the status of the environment by (iv) involving the movement of a participant in the space, e.g., a truck that reaches the loading area; (v) changing the environment, e.g., a human that closes a door or moves a box in the middle of a passage.

Therefore, the need to link business processes with the physical environment has become prominent since the literature still neglects to fully support the interplay between BPMN and the physical environment. Existing works dealing with the concept of the environment within BPMN [11,15,8] have drawbacks that highlight the need for a more comprehensive and integrated approach to model this interplay. Indeed, they often propose extensions to the BPMN graphical notation, and none of them give a formal approach to explicitly represent the environment and how BPMN elements relate to it.

To fill this gap, in this paper, we present an approach to bridge BPMN collaborations and the physical environment. Reasoning on multiple participants amplifies the interaction with the environment, while communication enables the sharing of environmental information. More in detail, the contributions of our work are as follows:

- we discuss *environmental aspects* relevant to the modeling and execution of business processes;
- we exploit *place graphs* enriched with *place attributes* to describe the physical environment and inject these concepts into the BPMN meta-model
- we define *environmental BPMN collaborations* without modifying and/or introducing new elements in the BPMN standard;
- we describe this interplay through a *formal operational semantics*.

We facilitate the understanding of the above concepts by resorting to a fire-extinguishing collaborative scenario.

The rest of this paper is organized as follows. Section 2 shows the interplay of BPMN collaborations and the environment. Section 3 introduces the formal semantics of environmental BPMN collaborations. Section 4 illustrates our formalization at work through the running example. Section 5 introduces related work, and Section 6 concludes the paper by discussing the approach.

## 2 Bridging BPMN and environment

In this section, we present our approach to link the BPMN meta-model to the physical environment. We also introduce a collaborative scenario to give a concrete vision of the problem; we use this scenario throughout the paper as a running example.

**Running example.** The running example concerns a fire-extinguishing collaboration in a student dormitory. The collaboration is performed by a *fire control system* and a *fire-fighting* autonomous robot. The fire control system is in charge of detecting fire in dormitory rooms through fire sensors and informing the extinguisher robot that will reach the fire to extinguish it. The dormitory, inhabited by students, comprises a kitchen,

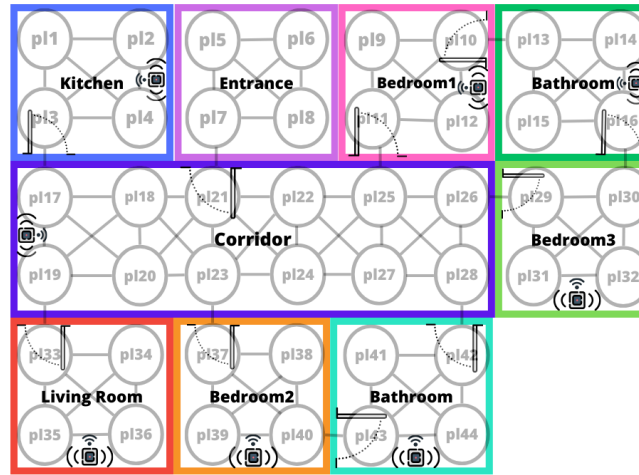


Fig. 1: Floor plan and place graph of the dormitory

an entrance, three bedrooms, two bathrooms, and a living room connected by a corridor. Each room has a door and a fire sensor. The collaboration starts when the fire control system detects the presence of fire in one of the dormitory rooms. In this case, the fire control system notifies the fire position to the fire-fighting robot and then activates the alarm. As soon as the robot receives the notification, it moves toward the fire and, after extinguishing it, sends feedback to the control system, which turns off the alarm. Notably, apart from the fire control system and the fire-fighting robot, some students in the dormitory perform their daily routines interacting with the environment.

**Model the environment.** To consider the environment in the modeling and execution of a business model, we need a way to represent it. Over the years, several models have been proposed to represent the physical environment [2]. In particular, two main classes of environmental models are defined: geometric models and symbolic models. The former mainly comprise cell-based and boundary-based geometrical representations. The latter uses topological-based structures or graphs, where it is possible to define an ad-hoc level of abstraction by capturing the connectivity and reachability between two locations in the space. Since our target is to link the environment to the BPMN notation, we chose place graphs as they can abstract the space topology at will, like also BPMN does [17,9]. Indeed, place graphs are a particular kind of symbolic model, where places symbolize a predefined area, like buildings and rooms, and the edges between two places represent connections that make it possible to move between them [2,7]. Thus, they provide a direct way to represent movements between places and planning routes to destinations. Moreover, place graphs can be enriched with attributes related to places (e.g., the illumination status) and edges (e.g., the distance).

Referring to the running example, Figure 1 depicts the place graph of the dormitory on top of its floor plan. To cover the entire dormitory, we spread the places on the floor plan homogeneously, i.e., adjacent places are equidistant and connected through edges, except for places divided by a wall or a closed door. Besides the representation

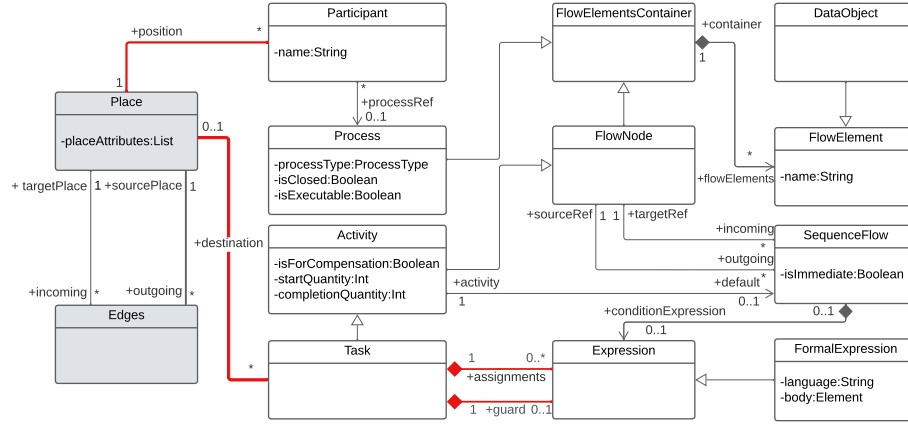


Fig. 2: BPMN meta-model extended with the environment

of the space, we include in the place graph the attributes necessary to handle the fire-extinguishing collaboration. In particular, we include in each place attributes indicating the presence of fire and the status of the alarm.

**Extending the BPMN meta-model.** To interconnect BPMN collaborations with place graphs, we extend the BPMN meta-model using its extensibility mechanism [16] as described in Figure 2.

We introduce two new classes, called *Place* and *Edge*, which respectively correspond to places and edges of the place graph. Places refer to edges (and vice-versa) through two attributes *sourcePlace* and *targetPlace*. Therefore, each place can be linked to many source/target places, permitting the representation of the entire place graph. Each place also has a list of attributes, which refers to physical environment characteristics. The connection between the environment and collaboration participants happens through the attribute *position* that links *Participant* and *Place* classes to indicate the current position of participants. Moreover, the *Task* class is linked to the *Place* class via the attribute *destination* that indicates the eventual target destination of an activity involving a movement in the space. Finally, we introduce the *guard* and *assignment* attributes to constrain the execution of a task to an environmental status and to modify the environment topology or its attributes.

To give a concrete view of the link between a BPMN collaboration and the environment, we provide in Figure 3 the collaboration diagram representing the behavior of the running example connected to the environmental model in Figure 1. The collaboration comprises a pool encapsulating the behavior of the fire control system and a pool referring to the fire-fighting robot. In addition, we include a pool that represents the behavior of a student in the dormitory. The student is not a direct participant in the collaboration, but it shares the same space and can modify the environment (i.e., activating the alarm, closing/opening doors) having consequences in the outcome of the collaboration. The fire-extinguishing collaboration process starts when the fire control system detects a fire via the conditional start event *Fire detected*. This element reacts to environmental changes; more precisely, it triggers a process instance when its condition

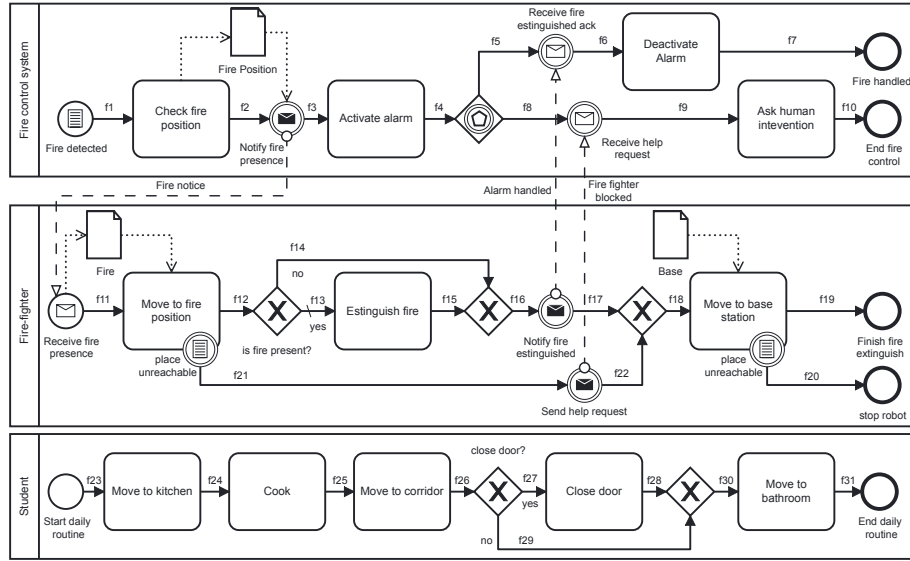


Fig. 3: Fire-extinguishing collaboration model

is evaluated as true. The expression predicates on the state of the fire sensors; in case a sensor reveals the fire, the condition becomes true. Therefore, the fire control system checks the position of the fire and notifies the fire-fighting robot using the message *fire notice*. Then, it activates the alarm through an assignment on the environmental attribute representing the status of the alarm. When the robot receives the notification with the message start event, it performs *Move to fire position*. This activity involves the movement of the process participant in the physical space. In particular, the robot will try to reach the destination (corresponding to the fire position) from its current location. If the robot manages to reach the destination, it extinguishes the fire if present, changes the status of the environment, and sends feedback to the fire control system, which turns off the alarm and ends its process. Then, the robot returns to the base station and stops. Regarding the *Move to fire position* and the *Move to base station* activities, if the robot cannot reach a destination, the conditional boundary events *place unreachable* trigger exceptional behaviours. The unreachability could be caused by the absence of a path from the source location to the destination. The boundary event attached to the *Move to fire position* activity leads the robot to notify its blocked condition to the fire control systems and return to the base station. The boundary associated with *Move to base station*, instead, stops the process.

### 3 Formal account of environmental BPMN collaborations

In this section, we formalize the semantics of BPMN collaborations including the environment, namely *environmental BPMN collaboration* models. To keep the formalization manageable and understandable, we focus on elements that are strictly needed

to define meaningful collaborations, the environment where the collaboration happens, and the relations between them.

### 3.1 Textual notation of environmental BPMN collaborations

The formalization resorts to a textual representation, defined in Figure 4, of BPMN models and of the environment. This is not a notation alternative to BPMN, but just a Backus-Naur Form (BNF) syntax of collaboration models (similar to the formalization proposed in [18]). Hence, the textual notation does not go beyond the expressiveness of the BPMN notation as given by the extended meta-model in Figure 2. The correspondence between the graphical notation of BPMN and the textual representation used here is straightforward, as highlighted by the description associated to each syntactical term in Figure 4; for a more detailed account of the one-to-one correspondence, the interested reader can refer to Appendix A. It is worth noticing that the BPMN standard deliberately leaves underspecified the expression language, whose instantiation is left to the designer to deal with specific concrete uses of the notation (common choices are the languages XPath and FEEL). In our formalization, we do not specify the expression language as well; however, we assume that expressions contain, at least, values, data objects and environmental attributes, and operations on the place graph.

The motivation for using here a textual representation rather than the usual graphical notation is that the former is more manageable for writing operational semantic rules than the latter. In addition, the graphical notation makes explicit those technical details of collaboration models that are not part of the graphical representation, but are part of the low-level XML characterization of the model. This information is needed to properly define the execution semantics of the models.

Figure 4 reports the BNF syntax defining the textual notation describing the structure of environmental BPMN collaboration models. Specifically, the upper part of the table reports the grammar productions defining the syntax of collaboration models, the middle part defines place graphs, and the bottom part reports the notation for the generic elements of the syntactic categories. Notably, even if this syntax would allow to write collaboration models that cannot be expressed in BPMN, here are considered only those terms of the syntax that can be derived from BPMN models. Intuitively, a collaboration is rendered in the presented syntax as a collection of single-instance pools, each one specifying a process, coupled with an environmental model in the form of a place graph. Formally, a collaboration  $C$  is a composition by means of the operator  $\parallel$ , of pool elements  $\text{pool}(p, P)$  uniquely identified by a pool name  $p$  and enveloping process structures of the form  $P$ . Similarly, a process  $P$  is a composition of process elements (denoted by the sans serif font) by means of the operator  $|$ . A place graph  $S$  is a tuple  $(P_l, E)$  where  $P_l$  is the set of places of interest for the collaboration and  $E$  is the set of edges linking couples of places. The union of the terms  $C$  and  $S$  composes the structure of an environmental BPMN collaboration model.

In Figure 5, we provide the textual representation of the environmental BPMN collaboration model of the running example. This exemplifies the correspondence between the BPMN graphical notation and the proposed BNF syntax. Moreover, we exploit the textual representation of the example Sec. 4 for discussing the semantics of the running scenario. In the textual representation, there is some information e.g., payload of

$C ::=$	$\text{pool}(p, P) \mid C \parallel C$	<i>(Collaboration Structures)</i> (pool, pool composition)
$P ::=$	$\text{start}(f, f') \mid \text{end}(f)$ $\mid \text{startRcv}(m : \text{do.a}, f)$ $\mid \text{startCond}(\text{exp}, f)$ $\mid \text{endSnd}(f, m : \text{exp})$ $\mid \text{interRcv}(f, m : \text{do.a}, f')$ $\mid \text{interSnd}(f, m : \text{exp}, f')$ $\mid \text{interCondCatch}(f, \text{exp}, f')$ $\mid \text{andSplit}(f, F) \mid \text{andJoin}(F, f)$ $\mid \text{xorSplit}(f, (\text{exp}_1, f_1), \dots, (\text{exp}_n, f_n))$ $\mid \text{xorJoin}(F, f)$ $\mid \text{eventBased}(f, (m_1 : \text{do.a}_1, f_1), \dots, (m_h : \text{do.a}_h, f_h),$ $\quad (\text{exp}_1, f_{h+1}), \dots, (\text{exp}_n, f_{h+n}))$ $\mid \text{task}(f, t, \text{exp}, A_A, A_S, D, B, f') \mid P \mid P$	<i>(Process Structures)</i> (start event, end event) (message receiving event) (conditional start event) (message sending end event) (msg receiving intermediate event) (msg sending intermediate event) (conditional catching interm. event) (AND split/join gateway) (XOR split gateway) (XOR join gateway) (event-based gateway) (task, process composition)
$A_A ::=$	$\text{do.a} := \text{exp}$ $\mid \text{pl.a} := \text{exp}$ $\mid \varepsilon \mid A_A, A_A$	<i>(Attribute Assignments)</i> (data object attribute assignment) (place attribute assignment) (empty, assignments list)
$A_S ::=$	$\text{connect}(e) \mid \text{disconnect}(e)$ $\mid \varepsilon \mid A_S, A_S$	<i>(Spatial Graph Actions)</i> (connect, disconnect) (empty, actions list)
$D ::=$	$\text{nil} \mid \text{pl} \mid \text{do.a}$	<i>(Destinations)</i> (empty, place, data object attribute)
$B ::=$	$\text{boundCond}(\text{exp}, f)$ $\mid \text{boundRcv}(m : \text{do.a}, f)$ $\mid \text{nil} \mid B, B$	<i>(Boundary Events)</i> (conditional boundary event) (message receiving boundary event) (empty, boundary events list)
$S ::=$	$(P_l, E) \quad \text{with } E \subseteq P_l \times P_l$	<i>(Place Graphs)</i>
<i>(Notation)</i>		
Pool name: $p$		Data object name: $\text{do}$
Sequence flow: $f$		Data object attribute: $\text{do.a}$
Sequence flow set: $F$		Place name: $\text{pl}$
Message flow: $m$		Place attribute: $\text{pl.a}$
Expression: $\text{exp}$		Place set: $P_l$
Value: $v$		Edge name: $e$
Task name: $t$		Edge set: $E$

Fig. 4: BNF syntax of environmental BPMN collaboration models.

$C_d = \text{pool}(p_{fcs}, P_{fcs}) \mid \text{pool}(p_f, P_f) \mid \text{pool}(p_s, P_s)$
$P_{fcs} = \text{startCond}(\text{exp}_1, f_1) \mid \text{task}(f_1, \text{Check fire position, true, Fire position.pos} := \text{pl}_j, \varepsilon, \text{nil, nil, } f_2) \mid$ $\text{interSnd}(f_2, \text{Fire notice: Fireposition.pos, } f_3) \mid \text{task}(f_3, \text{Activate alarm, true, } A_{A1}, \varepsilon, \text{nil, nil, } f_4) \mid$ $\text{eventBased}(f_4, (\text{Fire fighter blocked: true, } f_9), (\text{Alarm handled: true, } f_6)) \mid$ $\text{task}(f_6, \text{Deactivate alarm, true, } A_{A2}, \varepsilon, \text{nil, nil, } f_7) \mid$ $\text{task}(f_9, \text{Ask human intervention, true, } \varepsilon, \varepsilon, \text{nil, nil, } f_{10}) \mid \text{end}(f_7) \mid \text{end}(f_{10})$ $\text{exp}_1 = \bigvee_1^{44} \text{pl}_j.\text{fire} \quad A_{A1} = \forall 1 \leq j \leq 44 . \text{pl}_j.\text{alarm} := \text{true}$ $A_{A2} = \forall 1 \leq j \leq 44 . \text{pl}_j.\text{alarm} := \text{false}$
$P_f = \text{startRcv}(\text{Fire notice: Fire.pos, } f_{11}) \mid$ $\text{task}(f_{11}, \text{Move to fire position, true, } \varepsilon, \varepsilon, \text{Fire.pos, boundCond}(\text{exp}_4, f_{21}), f_{12}) \mid$ $\text{xorSplit}(f_{12}, (\text{Fire.pos.fire} = \text{true, } f_{13}), (\text{Fire.pos.fire} = \text{false, } f_{14})) \mid$ $\text{task}(f_{13}, \text{Extinguish fire, true, Fire.pos.fire} := \text{false, } \varepsilon, \text{nil, nil, } f_{15}) \mid$ $\text{xorJoin}((f_{14}, f_{15}), f_{16}) \mid \text{interSnd}(f_{16}, \text{Alarm handled: true, } f_{17}) \mid$ $\text{interSnd}(f_{21}, \text{Fire fighter blocked: true, } f_{22}) \mid \text{xorJoin}((f_{17}, f_{22}), f_{18}) \mid$ $\text{task}(f_{18}, \text{Move to base station, true, } \varepsilon, \varepsilon, \text{Base.pos, boundCond}(\text{exp}_4, f_{20}), f_{19}) \mid \text{end}(f_{19}) \mid \text{end}(f_{20})$ $\text{exp}_4 = \text{next}(S, \sigma_p, p, D) = \emptyset$
$P_s = \text{start}(f'_{23}, f_{23}) \mid \text{task}(f_{23}, \text{Move to kitchen, true, } \varepsilon, \varepsilon, \text{pl}_4, \text{nil, } f_{24}) \mid$ $\text{task}(f_{24}, \text{Cook, true, pl}_4.\text{fire} := \text{true, } \varepsilon, \text{nil, nil, } f_{25}) \mid \text{task}(f_{25}, \text{Move to corridor, true, } \varepsilon, \varepsilon, \text{pl}_{17}, \text{nil, } f_{26}) \mid$ $\text{xorSplit}(f_{26}, (\text{true, } f_{27}), (\text{true, } f_{29})) \mid \text{task}(f_{27}, \text{Close door, true, } \varepsilon, A_{S1}, \text{nil, nil, } f_{28}) \mid$ $\text{xorJoin}((f_{28}, f_{29}), f_{30}) \mid \text{task}(f_{30}, \text{Move to bathroom, true, } \varepsilon, \varepsilon, \text{pl}_{15}, \text{nil, } f_{31}) \mid \text{end}(f_{31})$ $A_{S1} = \text{disconnect}((\text{pl}_{17}, \text{pl}_3)), \text{disconnect}((\text{pl}_3, \text{pl}_7))$
$S_d = (P_d, E_d) \quad P_d = \{\text{pl}_j \mid 1 \leq j \leq 44\} \quad E_d = \{(\text{pl}_1, \text{pl}_2), (\text{pl}_1, \text{pl}_3), (\text{pl}_1, \text{pl}_4), (\text{pl}_2, \text{pl}_1), (\text{pl}_2, \text{pl}_3), (\text{pl}_2, \text{pl}_4), \dots\}$

Fig. 5: Textual representation of the running example.

message events, assignments, guard conditions, and locations that is not reported in the graphical notation but is reported in the textual representation since it is needed to properly define the execution semantics of the environmental BPMN collaborations. Moreover, each sequence flow in the graphical notation is split into two parts: the part outgoing from the source element and the part incoming into the target element. The two parts are correlated by a unique sequence flow name. We do not provide a direct syntactic representation of data objects and place attributes, the evolution of their state during the model execution is a semantic concern described later in this section.

We assume a generic record structure for data objects and place attributes: data objects and places are just a list of attributes characterized by a name and the corresponding value. Specifically, the attribute name of the data object *do* (resp. place *pl*) is accessed via the notation *do.a* (resp. *pl.a*). Since we explicitly consider data and place attributes, tasks, XOR split gateways, conditional events, and messages can use them. Tasks can express guard conditions that predicate these attributes via conditional expressions. They also express a list of attribute assignments  $A_A$  that assign values (resulting from the evaluation of expressions) to data object or place attributes, e.g., *do.a* := *v*; and a list of spatial assignment  $A_S$  that modify the space topology by removing or adding a connection between two places, e.g., *disconnect(e)*. Finally, tasks may involve a movement in the place graph; in this case, the destination *D* identifies one of the places in the place graph. XOR split gateways specify conditions in their outgoing edge to decide which sequence flow to activate. This is formally rendered mapping sequence flows to conditional expressions, e.g., (*exp*, *f*). Notably, we assume that the set of expressions includes the distinguished expression *default* referring to the default sequence flow. Conditional events are characterized by conditional expressions in the form *exp*, which are evaluated to trigger the event. Finally, message events describe point-to-point communications that may carry data. Throwing message events are characterized by the pair *m* : *exp*, where *m* is the (unique) message name and *exp* is an expression representing the payload. Catching message events are characterized by the



pair  $m : do.a$  where  $do.a$  denotes the data object attribute to which to assign the evaluated expression. To express communication without passing information we assume both throwing and catching events specify in the expression the value true.

### 3.2 Semantics of environmental BPMN collaborations

So far, the syntax represents the mere structure of processes, collaborations, and the environment. To describe the semantics, the structural information is enriched with a notion of execution state, given by the marking of sequence flows with tokens [10, p. 27], the value of data object and place attributes, the status of tasks, the position of the participants in the place graph, and payloads stored in the message queues.

These stateful descriptions are called process configurations and collaboration configurations. A *process configuration* (reps. *collaboration configuration*) has the form  $\langle P, S, \sigma_f, \sigma_a, \sigma_t, \sigma_p, \sigma_m \rangle$  (resp.  $\langle C, S, \sigma_f, \sigma_a, \sigma_t, \sigma_p, \sigma_m \rangle$ ) where  $P$  is a process structure (resp.  $C$  is a collaboration structure);  $S$  is the place graph;  $\sigma_f : \mathbb{F} \rightarrow \mathbb{N}$  is a *sequence flow state function* (where  $\mathbb{F}$  is the set of sequence flows and  $\mathbb{N}$  is the set of natural numbers) specifying for each sequence flow the current number of tokens marking it;  $\sigma_a : \mathbb{A} \rightarrow \mathbb{V}$  is a *data and place state function* (where  $\mathbb{A}$  is the set of data and place attributes and  $\mathbb{V}$  is the set of values) assigning values (possibly null) to data object and place attributes;  $\sigma_t : \mathbb{T} \rightarrow \mathbb{N}$  is a *task state function* (where  $\mathbb{T}$  is the set of task names) used to keep track of the number of active task instances; the status of a task depends on its evolution from inactive to active state, where the inactive status formally corresponds to having zero active instances;  $\sigma_p : \mathbb{P} \rightarrow \mathbb{P}_l$  is the *place state function* (where  $\mathbb{P}$  is the set of pool names and  $\mathbb{P}_l$  is the set of places) mapping each collaboration participant to a place;  $\sigma_m : \mathbb{M} \rightarrow 2^{\mathbb{V}^n}$  is a *message state function* (where  $\mathbb{M}$  is the set of message flows) that assigns to each message flow name  $m$  an ordered multiset of values (i.e., a queue) representing the messages received along  $m$ .

With  $\sigma_f^0$  (resp.  $\sigma_a^0$ ,  $\sigma_t^0$ ,  $\sigma_p^0$ , and  $\sigma_m^0$ ) is denoted the sequence flow (resp. data and place attributes, task, place, and messages queue) state where all sequence flows are unmarked (resp. all data and place attributes are set to null, all tasks are inactive, places are empty, and messages queues are empty). Formally,  $\sigma_f^0(f) = 0 \ \forall f \in \mathbb{F}$ ,  $\sigma_a^0(do.a) = \text{null} \ \forall do.a \in \mathbb{A} \wedge \sigma_a^0(pl.a) = \text{null} \ \forall pl.a \in \mathbb{A}$ ,  $\sigma_t^0(t) = 0 \ \forall t \in \mathbb{T}$ ,  $\sigma_p^0(pl) = \text{null} \ \forall pl \in \mathbb{P}_l$ ,  $\sigma_m^0(m) = \emptyset \ \forall m \in \mathbb{M}$ . The state obtained by updating in  $\sigma_f$  the number of tokens of the sequence flow  $f$  to  $n$ , written as  $\sigma_f \cdot [f \mapsto n]$ , is defined as follows:  $(\sigma_f \cdot [f \mapsto n])(f')$  returns  $n$  if  $f' = f$ , otherwise it returns  $\sigma_f(f')$ . The updates of states  $\sigma_a$ ,  $\sigma_t$ ,  $\sigma_p$ , and  $\sigma_m$  are defined similarly.

The operational semantic is defined employing *labeled transition systems* (LTSs). The LTS of the process behavior is a triple  $\langle \mathcal{P}, \mathcal{L}, \rightarrow \rangle$  where:  $\mathcal{P}$  is a set of processes configurations;  $\mathcal{L}$ , ranged over by  $l$ , is a set of labels; and  $\rightarrow \subseteq \mathcal{P} \times \mathcal{L} \times \mathcal{P}$  is a transition relation. As simplification,  $\langle P, S, \sigma_f, \sigma_a, \sigma_t, \sigma_p, \sigma_m \rangle \xrightarrow{l} \langle P, S, \sigma'_f, \sigma'_a, \sigma'_t, \sigma'_p, \sigma'_m \rangle$  indicates that  $(\langle P, S, \sigma_f, \sigma_a, \sigma_t, \sigma_p, \sigma_m \rangle, l, \langle P, S, \sigma'_f, \sigma'_a, \sigma'_t, \sigma'_p, \sigma'_m \rangle) \in \rightarrow$ , and says that ‘the process in the configuration  $\langle P, S, \sigma_f, \sigma_a, \sigma_t, \sigma_p, \sigma_m \rangle$  can do a transition labelled by  $l$  and evolve to the process configuration  $\langle P, S, \sigma'_f, \sigma'_a, \sigma'_t, \sigma'_p, \sigma'_m \rangle$ . The LTS describing the behavior of environmental BPMN collaborations is a triple  $\langle \mathcal{C}, \mathcal{L}, \rightarrow \rangle$  where:  $\mathcal{C}$  is a set of collaboration configurations, and  $\rightarrow \subseteq \mathcal{C} \times \mathcal{L} \times \mathcal{C}$  is a transition relation. The labels

Functions $inc(\sigma_f, f) = \sigma_f \cdot [f \mapsto \sigma_f(f) + 1]$ , resp. $dec(\sigma_f, f) = \sigma_f \cdot [f \mapsto \sigma_f(f) - 1]$ , increments/decrements by one the number of tokens marking the sequence flow $f$ in the state $\sigma_f$ . They extend in a natural way the sets $F$ of sequence flows. Specifically, they are inductively defined as follows: $inc(\sigma_f, \emptyset) = dec(\sigma_f, \emptyset) = \sigma_f$ , $inc(\sigma_f, \{f\} \cup F) = inc(inc(\sigma_f, f), F)$ , resp. $dec(\sigma_f, \{f\} \cup F) = dec(dec(\sigma_f, f), F)$ .
Functions $inc(\sigma_t, t)$ , respectively $dec(\sigma_t, t)$ , activate/deactivate the task $t$ in the state $\sigma_t$ . Their definitions are similar to the ones of functions $inc(\sigma_f, f)$ and $dec(\sigma_f, f)$ .
Relation $eval(exp, \sigma_a, v)$ states that $v$ is one of the possible values resulting from the evaluation of the expression $exp$ on the data and place state $\sigma_a$ ; this is a relation, because an expression may contain non-deterministic operators, and is not explicitly defined, since the syntax of expressions is deliberately not specified.
Function $upd(\sigma_a, A_A)$ performs the attribute assignments $A_A$ in the state $\sigma_a$ . It is inductively defined as follows: $upd(\sigma_a, e) = \sigma_a$ ; $upd(\sigma_a, do.a := v) = \sigma_a \cdot [do.a \mapsto v]$ ; $upd(\sigma_a, pl.a := exp) = \sigma_a \cdot [pl.a \mapsto v]$ ; $upd(\sigma_a, (A_{A1}, A_{A2})) = \sigma_a''$ with $\sigma_a'' = upd(\sigma_a', A_{A2})$ and $\sigma_a' = upd(\sigma_a, A_{A1})$ .
Function $mod(S, A_S)$ performs the spatial assignments $A_S$ in the place graph $S$ . It is inductively defined as follows: $upd(S, e) = S$ ; $upd(S, connect(e)) = S \cdot [e \mapsto E + \{e\}]$ ; $upd(S, disconnect(e)) = S \cdot [e \mapsto E - \{e\}]$ .
Function $next(S, \sigma_p, p, D)$ returns the set of places that are next in the shortest paths in $S = (P, E)$ from the current place of the process $\sigma_p(p)$ to the destination $\sigma_a(D)$ (with abuse of notation, here we assume that $\sigma_a(pl) = pl$ for any $pl$ ). It is inductively defined as follows: $next(S, \sigma_p, p, D) = \emptyset$ if $D = nil$ , or it does not exist a path from $\sigma_p(p)$ to $\sigma_a(D)$ , or $\sigma_p(p) = \sigma_a(D)$ ; $next(S, \sigma_p, p, D) = \{pl \mid pl \text{ belongs to a path from } \sigma_p(p) \text{ to } \sigma_a(D) \text{ and } (\sigma_p(p), pl) \in E\}$ .
Function $move(\sigma_p, p, pl) = \sigma_p \cdot [p \mapsto pl]$ assigns to the process $p$ the place $pl$ in the state $\sigma_p$ .
Relation $whichActive(B, \sigma_a, \sigma_m, f^b)$ states that $f^b$ is the sequence flow of one of the possible active boundary event in $B$ . It is inductively defined as follows: $whichActive(nil, \sigma_a, \sigma_m, nil)$ ; $whichActive(sigBound(m: do.a, f^b) \mid B, \sigma_a, \sigma_m, f^b) \iff \sigma_m(m) \neq \emptyset$ ; $whichActive(boundCond(exp, f^b) \mid B, \sigma_a, \sigma_m, f^b) \iff eval(exp, \sigma_a, true)$ .
Function $set(\sigma_p, p, pl) = \sigma_p \cdot [p \mapsto pl]$ sets the position of process $p$ to the location $pl$ .
Function $add(\sigma_m, m, v) = \sigma_m \cdot [m \mapsto \sigma_m(m) + \{v\}]$ adds the value $v$ for the message flow $m$ in the state $\sigma_m$ .
Function $rm(\sigma_m, m) = \sigma_m \cdot [m \mapsto \sigma_m(m) - \{v\}]$ removes the message $m$ from the state $\sigma_m$ .

Fig. 6: Auxiliary functions and relations

used by the transition relations  $\rightarrow$  are generated by  $l ::= \checkmark \mid \tau$ , where  $\checkmark$  denotes the passing of a unit of time (due, e.g., to movements in the place graph), and  $\tau$  denotes a silent transition (due, e.g., token flow in the processes).

To improve the readability, the following simplifications reduce the notation of operational rules. Specifically, unnecessary information is omitted: (i) the states  $\sigma_f$ ,  $\sigma_a$ ,  $\sigma_t$ ,  $\sigma_p$ , and  $\sigma_m$  (when not used) from the source configuration of transitions; (ii) the collaboration, the place graph, and the states not affected by transitions, from the target configuration. Thus, for example, a transition  $\langle P, S, \sigma_f, \sigma_a, \sigma_t, \sigma_p, \sigma_m \rangle \xrightarrow{l} \langle P, S, \sigma'_f, \sigma'_a, \sigma'_t, \sigma'_p, \sigma'_m \rangle$  will be written as  $P \xrightarrow{l} \langle \sigma'_f \rangle$  when it simply affects the sequence flow state function. The same simplification holds for a collaboration  $C$ .

To simplify the definition of the operational rules, in Figure 6, we define *auxiliary functions and relations* that update states of process configurations and act on the place graph topology. The operational rules defining the transition relation of the process semantics are given by the inference rules in Figures 7, and 8. We now briefly comment on these rules starting from those dealing with events and gateways in Figures 7. Rule *P-Start* starts the execution of a process when it has been activated. To denote the enabled status of start events we have included in their syntactical definition an incoming (spurious) sequence flow, named enabling sequence flow. Thus, the process is activated when the enabling sequence flow of a start event is marked. The effect of the rule is to increment the number of tokens in the sequence flow outgoing from the start event and to decrease the marking of the enabling sequence flow. To show how the auxiliary functions and relations of Figure 6 update process configurations, we exemplify their outcome in the application of rule *P-Start*, without applying the simplifications to

$\text{start}(f, f') \xrightarrow{\tau} \langle \text{inc}(\text{dec}(\sigma_f, f), f') \rangle$	$\sigma_f(f) > 0$	(P-Start)
$\text{end}(f) \xrightarrow{\tau} \langle \text{dec}(\sigma_f, f) \rangle$	$\sigma_f(f) > 0$	(P-End)
$\langle \text{startRcv}(m : \text{do.a}, f), \sigma_f^0, \sigma_t^0 \rangle \xrightarrow{\tau}$ $\langle \text{inc}(\sigma_f, f), \text{upd}(\sigma_a, \text{do.a} := v), \text{rm}(\sigma_m, m) \rangle$	$v \in \sigma_m(m)$	(P-StartRcv)
$\langle \text{startCond}(\text{exp}, f), \sigma_f^0, \sigma_t^0 \rangle \xrightarrow{\tau} \langle \text{inc}(\sigma_f, f) \rangle$	$\text{eval}(\text{exp}, \sigma_a, \text{true})$	(P-StartCond)
$\text{endSnd}(f, m : \text{exp}) \xrightarrow{\tau} \langle \text{dec}(\sigma_f, f), \text{add}(\sigma_m, m, v) \rangle$	$\sigma_f(f) > 0 \wedge$ $\text{eval}(\text{exp}, \sigma_a, v)$	(P-EndSnd)
$\text{interRcv}(f, m : \text{do.a}, f') \xrightarrow{\tau}$ $\langle \text{inc}(\text{dec}(\sigma_f, f), f'), \text{upd}(\sigma_a, \text{do.a} := v), \text{rm}(\sigma_m, m) \rangle$	$\sigma_f(f) > 0 \wedge$ $v \in \sigma_m(m)$	(P-InterRcv)
$\text{interCondCatch}(f, \text{exp}, f') \xrightarrow{\tau} \langle \text{inc}(\text{dec}(\sigma_f, f), f') \rangle$	$\sigma_f(f) > 0 \wedge$ $\text{eval}(\text{exp}, \sigma_a, \text{true})$	(P-CondCatch)
$\text{interSnd}(f, m : \text{exp}, f') \xrightarrow{\tau}$ $\langle \text{inc}(\text{dec}(\sigma_f, f), f'), \text{add}(\sigma_m, m, v) \rangle$	$\sigma_f(f) > 0 \wedge$ $\text{eval}(\text{exp}, \sigma_a, v)$	(P-InterSnd)
$\text{andSplit}(f, F) \xrightarrow{\tau} \langle \text{inc}(\text{dec}(\sigma_f, f), F) \rangle$	$\sigma_f(f) > 0$	(P-AndSplit)
$\text{xorSplit}(f, (\text{exp}_1, f_1), \dots, (\text{exp}_n, f_n)) \xrightarrow{\tau}$ $\langle \text{inc}(\text{dec}(\sigma_f, f), f') \rangle$	$\sigma_f(f) > 0 \wedge$ $\exists 1 \leq j \leq n.$	(P-XorSplit <sub>1</sub> )
$\text{xorSplit}(f, (\text{exp}_1, f_1), \dots, (\text{default}, f'), (\text{exp}_n, f_n)) \xrightarrow{\tau}$ $\langle \text{inc}(\text{dec}(\sigma_f, f), f') \rangle$	$\sigma_f(f) > 0 \wedge$ $\forall 1 \leq j \leq n.$	(P-XorSplit <sub>2</sub> )
$\text{andJoin}(F, f) \xrightarrow{\tau} \langle \text{inc}(\text{dec}(\sigma_f, F), f) \rangle$	$\forall f' \in F.$ $\sigma_f(f') > 0$	(P-AndJoin)
$\text{xorJoin}(\{f\} \cup F, f') \xrightarrow{\tau} \langle \text{inc}(\text{dec}(\sigma_f, f), f') \rangle$	$\sigma_f(f) > 0$	(P-XorJoin)
$\text{eventBased}(f, (m_1 : \text{do.a}_1, f_1), \dots, (m_h : \text{do.a}_h, f_h),$ $(\text{exp}_{h+1}, f_{h+1}), \dots, (\text{exp}_{h+n}, f_{h+n})) \xrightarrow{\tau}$ $\langle \text{inc}(\text{dec}(\sigma_f, f), f_j) \rangle$	$\sigma_f(f) > 0 \wedge$ $\exists h+1 \leq j \leq h+n.$	(P-EventGC)
$\text{eventBased}(f, (m_1 : \text{do.a}_1, f_1), \dots, (m_h : \text{do.a}_h, f_h),$ $(\text{exp}_{h+1}, f_{h+1}), \dots, (\text{exp}_{h+n}, f_{h+n})) \xrightarrow{\tau}$ $\langle \text{inc}(\text{dec}(\sigma_f, f), f_j) \rangle$	$\sigma_f(f) > 0 \wedge$ $\exists 1 \leq j \leq n.$	(P-EventGM)
$\langle \text{inc}(\text{dec}(\sigma_f, f), f_j), \text{upd}(\sigma_a, \text{do.a} := v), \text{rm}(\sigma_m, m_j) \rangle$	$v \in \sigma_m(m_j)$	

Fig. 7: BPMN process semantics: events and gateways.

the transition. Let us assume that  $\sigma_f(f)$  is 1 for  $f = f_1$  and 0 for any other  $f$ ; thus  $\langle \text{start}(f_1, f_2), S, \sigma_f, \sigma_a, \sigma_t, \sigma_p, \sigma_m \rangle \xrightarrow{\tau} \langle \text{start}(f_1, f_2), S, \sigma'_f, \sigma_a, \sigma_t, \sigma_p, \sigma_m \rangle$  where  $\sigma'_f(f)$  is 1 for  $f = f_2$  and 0 for any other  $f$ . Indeed,  $\text{dec}(\sigma_f, f_1) = \sigma_f \cdot [f_1 \mapsto \sigma_f(f_1) - 1] = \sigma_f \cdot [f_1 \mapsto 0] = \sigma'_f$  and, hence,  $\sigma'_f = \text{inc}(\text{dec}(\sigma_f, f_1), f_2) = \text{inc}(\sigma'_f, f_2) = \sigma'_f \cdot [f_2 \mapsto \sigma'_f(f_2) + 1] = \sigma'_f \cdot [f_2 \mapsto 1]$ ; thus, we have  $\sigma'_f = \sigma_f \cdot [f_1 \mapsto 0] \cdot [f_2 \mapsto 1]$ .

Rule *P-End* instead is enabled when there is at least one token in the incoming sequence flow of the end event, which is then consumed. Rule *P-StartRcv* starts the process execution when there is a value in the message queue, then it increments the number of tokens in the outgoing sequence flow, updates the data object with the received value, and removes it from the queue. Rule *P-StartCond* starts the execution of a process when the conditional expression is evaluated as true, incrementing the number of tokens in the sequence flow. Rule *P-EndSnd* is enabled when there is at least a token

in the incoming edge of the end event, which is then removed. Moreover, the value resulting from the evaluation of the expression  $\text{exp}$  is added to the message queue. Rules  $P\text{-InterRcv}$  and  $P\text{-InterSnd}$  are enabled when there is at least a token in their incoming edge, then behave respectively as rules  $P\text{-StartRcv}$  and  $P\text{-EndSnd}$ . Rules  $P\text{-CondCatch}$  is enabled when there is at least a token in their incoming edge, then behaves as rules  $P\text{-StartCond}$ . Rule  $P\text{-AndSplit}$  is applied when there is at least one token in the incoming edge of an AND split gateway; as a result of its application, the rule decrements the number of tokens in the incoming sequence flow, and increments the tokens in each outgoing sequence flow. Rule  $P\text{-XorSplit1}$  is applied when a token is available in the incoming sequence flow of a XOR split gateway and a conditional expression of one of its outgoing sequence flows is evaluated to true; the rule decrements the token in the incoming sequence flow and increments the token in the selected outgoing sequence flow. Notably, if more edges have their guards satisfied, one of them is non-deterministically chosen. Rule  $P\text{-XorSplit2}$  is applied when all guard expressions are evaluated to false; in this case, the default sequence flow is marked. Rule  $P\text{-AndJoin}$  decrements the tokens in each incoming sequence flow and increments the number of tokens of the outgoing sequence flow, when each incoming sequence flow has at least one token. Rule  $P\text{-XorJoin}$  is activated every time there is a token in one of the incoming sequence flows, which is then moved to the outgoing sequence flow. Rules  $P\text{-EventG}_C$  and  $P\text{-EventG}_S$  are activated when there is a token in the incoming edge and respectively a conditional expression is evaluated as true or a value is present in a message queue. The application of the rule moves the token from the incoming edge to the outgoing edge corresponding to the true condition ( $P\text{-EventG}_C$ ) or the non-empty queue ( $P\text{-EventG}_M$ ). In the latter rule, the data object attribute is updated and the value is removed from the message queue.

Rules in Figure 8 deal with task execution following the task status evolution from inactive ( $\sigma_i(t) = 0$ ), to active ( $\sigma_i(t) = 1$ ); and the interleaving of processes and collaboration elements. Rule  $P\text{-Task}$  activates the task when it is not already active, there is a token in the incoming edge, and the guard  $\text{exp}$  is satisfied. Then, it removes the token in the incoming sequence flow and increments the task state to notify its activation. Rule  $P\text{-Task}_M$  deals with tasks that involve a movement in the place graph, i.e., when the destination  $D$  is defined. If the task is active, but not its eventual boundary events, and the destination is not reached (the *next* function returns a non-empty set), the rule moves the process participant from the current position to one of the next places in the path toward the destination place. Since movements in space consume time in reality, this rule produces a transition labeled with  $\checkmark$ . Rule  $P\text{-Task}_A$  deals with task assignments. If the task is active, but not its eventual boundary events, and either the destination is nil or reached, the rule increments the outgoing sequence flow, performs the assignments, and deactivates the task. Rule  $P\text{-Task}_B$  deals with the boundary events  $B$  attached to the task. If the task and one of the boundary events are active, the rule adds a token in the outgoing sequence flow of the boundary event and deactivates the task. Regarding the operational rules dealing with the interleaving of process and collaboration elements, rule  $P\text{-}\tau$  enables single process elements to evolve the process configuration with silent transitions  $\tau$ . Instead, rules  $P\text{-Task}_E$ ,  $P\text{-Empty}$ , and  $P\text{-}\checkmark$  enable process elements to evolve the process configuration with temporal transitions  $\checkmark$ . Regarding collaboration elements, rule  $C\text{-Int}$  propagates the transition from process to collaboration configura-

$\text{task}(f, t, \text{exp}, A, D, B, f') \xrightarrow{\tau} \langle \text{dec}(\sigma_f, f), \text{inc}(\sigma_t, t) \rangle$	$\sigma_f(f) > 0 \wedge \sigma_t(t) = 0$ $\wedge \text{eval}(\text{exp}, \sigma_a, \text{true})$	(P-Task)
$\text{task}(f, t, \text{exp}, A, D, B, f') \xrightarrow{\checkmark} \langle \text{move}(\sigma_p, p, pl) \rangle$	$\sigma_t(t) > 0 \wedge pl \in \text{next}(S, \sigma_p, p, D)$ $\wedge \text{whichActive}(B, \sigma_a, \sigma_m, \text{nil})$	(P-Task <sub>M</sub> )
$\text{task}(f, t, \text{exp}, A_A, A_S, D, B, f') \xrightarrow{\tau} \langle \text{mod}(S, A_S), \text{inc}(\sigma_f, f'), \text{upd}(\sigma_a, A_A), \text{dec}(\sigma_t, t) \rangle$	$\sigma_t(t) > 0 \wedge \text{next}(S, \sigma_p, p, D) = \emptyset$ $\wedge \text{whichActive}(B, \sigma_a, \sigma_m, \text{nil})$	(P-Task <sub>A</sub> )
$\text{task}(f, t, \text{exp}, A, D, B, f') \xrightarrow{\tau} \langle \text{inc}(\sigma_f, f^b), \text{dec}(\sigma_t, t) \rangle$	$\sigma_t(t) > 0$ $\wedge \text{whichActive}(B, \sigma_a, \sigma_m, f^b)$	(P-Task <sub>B</sub> )
$\text{task}(f, t, \text{exp}, A, D, B, f') \xrightarrow{\checkmark} \langle \rangle$	$\sigma_f(f) = 0 \wedge \sigma_t(t) = 0$	(P-Task <sub>E</sub> )
$P \xrightarrow{\checkmark} \langle \rangle$	$P \neq \text{task}(f, t, \text{exp}, A, D, B, f') \mid P'$	(P-Empty)
$\frac{P_1 \xrightarrow{\tau} \langle \sigma' \rangle}{P_1 \mid P_2 \xrightarrow{\tau} \langle \sigma' \rangle} \quad (P-\tau)$	$\frac{P_1 \xrightarrow{\checkmark} \langle \sigma_1 \rangle \quad \langle P_2, \sigma_1 \rangle \xrightarrow{\checkmark} \langle \sigma' \rangle}{P_1 \mid P_2 \xrightarrow{\checkmark} \langle \sigma' \rangle} \quad (P-\checkmark)$	
$\frac{P \xrightarrow{I} \langle \sigma' \rangle}{\text{pool}(p, P) \xrightarrow{I} \langle \sigma' \rangle} \quad (C-Int)$	$\frac{C_1 \xrightarrow{\tau} \langle \sigma_1 \rangle}{C_1 \parallel C_2 \xrightarrow{\tau} \langle \sigma' \rangle} \quad (C-\tau)$	$\frac{C_1 \xrightarrow{\checkmark} \langle \sigma_1 \rangle \quad \langle C_2, \sigma_1 \rangle \xrightarrow{\checkmark} \langle \sigma' \rangle}{C_1 \parallel C_2 \xrightarrow{\checkmark} \langle \sigma' \rangle} \quad (C-\checkmark)$

Fig. 8: BPMN process and collaboration semantics: tasks and interleaving.

tion, while  $C-\tau$  and  $C-\checkmark$  behave as  $P-\tau$  and  $P-\checkmark$  respectively. Notably, rules  $P-\tau$  and  $C-\tau$  allow single process elements to evolve the collaboration configuration, while  $P-\checkmark$  and  $C-\checkmark$  imposes a synchronous triggering of rules producing a temporal transition.

#### 4 Environmental BPMN collaboration formalization at work

In this section, we assess our formalization and show in practice the interplay between BPMN and the physical environment using the running example.

Referring to Figure 5, the fire-extinguishing collaboration structure is  $C_{dorm} = \text{pool}(p_{fcs}, P_{fcs}) \mid \text{pool}(p_f, P_f) \mid \text{pool}(p_s, P_s)$ , and the space model is  $S_d := (P_l, E)$ . The initial configuration of the environmental BPMN collaboration model is the following:  $\langle C_d, S_d, \sigma_f, \sigma_a^0, \sigma_t^0, \sigma_p, \sigma_m^0 \rangle$ , where  $\sigma_f = \sigma_f^0 \cdot [f'_{23} \mapsto 1]$  and  $\sigma_p = \sigma_p^0 \cdot [p_f \mapsto pl_{28}] \cdot [p_s \mapsto pl_{39}]$ . Following, we describe two different execution outcomes of the running example. Note that, we focus on the execution steps that better show the interaction between BPMN and the environment.

The collaboration starts with the enactment of the *Student*'s process since the only token in the collaboration is in  $f'_{23}$ . This enables rule  $P-Start$  which consumes the token in  $f'_{23}$  and produces a token in  $f_{23}$ . Then, the token in  $f_{23}$  enables rule  $P-Task$  on task *Move to kitchen*. As a consequence, rule  $P-Task$  decrements the sequence flow  $f_{23}$  and puts the task in the active status. Now that the task is active and the destination field is not empty, rule  $P-Task_M$  moves the process from its current place ( $pl_{39}$ ) toward the one corresponding to the kitchen ( $pl_4$ ) as reported in the corresponding syntax term. Thus,

$P\text{-Task}_M$  is repeated until the student reaches the kitchen, then rule  $P\text{-Task}_A$  deactivates the task and produces a token on  $f_{24}$  since no assignment is involved. At this point, rule  $P\text{-Task}$  activates activity *Cook*. This activity simulates the fact that the student starts the fire, this is formalized through the assignment  $pl_4.\text{fire} := \text{true}$  which sets as *true* the attribute *fire* in the location  $pl_4$ . Immediately after, the students perform task *Move to corridor* to reach the place  $pl_{17}$  using the same rules as for *Move to kitchen*. Here, the student can close or not the door linking the kitchen with the rest of the dormitory. Formally, the conditional expression in the sequence flows outgoing the XOR split gateway are both set to *true*, therefore whether a token is on  $f_{26}$ , rule  $P\text{-XorSplit}_1$  is enabled and it can non deterministically choose to increment the token on  $f_{27}$  or  $f_{29}$ .

This choice opens the way to two possible outcomes. The first is when the student leaves the door open. After the student sets the fire in the kitchen, the location attribute  $pl_4.\text{fire}$  changes to *true*. Therefore, rule  $P\text{-StartCond}$  is enabled for the conditional start event of the fire control system that notifies the fire-fighter of the fire position  $pl_4$ . Applying rule  $P\text{-Task}$ , the fire-fighter activates task *Move to fire position* and starts moving from its current position  $pl_{28}$  toward place  $pl_4$ . Essentially, the fire-fighter reaches the fire position and extinguishes it leading to the expected completion of the collaboration.

The latter case is when the student closes the door before moving toward its room. Indeed activity *Close the door* brings the assignments  $\text{disable}(pl_3, pl_{17})$  and  $\text{disable}(pl_{17}, pl_3)$  which removes the connections between places  $pl_3$  and  $pl_{17}$ . Again, the place attribute  $pl_4.\text{fire}$  is *true* therefore, the fire control system and the fire-fighter processes start. According to our semantics, which synchronizes temporal actions, rule  $P\text{-Task}_M$  starts to be applied at the same time on tasks *Move to corridor* and *Move to fire position*, immediately after the completion of task *Cook*. Therefore, to extinguish the fire, the fire-fighter has to reach the fire position before the student closes the door. The fire-fighter has to traverse at least seven edges to reach the kitchen from place  $pl_{28}$ , while the student moves from  $pl_4$  (previously reached by task *Move to kitchen*) and can reach place  $pl_{17}$  traversing only two edges. Since each step between adjacent places corresponds to the passing of a tick and the steps are synchronized in the whole collaboration (see rules  $P\text{-Task}_M$ ,  $P\text{-}\checkmark$  and  $C\text{-}\checkmark$ ), after two ticks from the start of the fire, the student closes the door, and the fire-fighter reaches  $pl_{22}$ . Then, the student continues moving toward his room, while  $P\text{-Task}_M$  does not produce effects for the task *Move to fire position*. This is because no more paths exist for reaching the kitchen, formally  $\text{next}(S, \sigma_p, p_f, pl_4) = \emptyset$ . Nonetheless, the conditional boundary event attached to the task is regulated by the conditional expression thus, once the door is closed the rule  $P\text{-Task}_B$  becomes active and the exceptional flow  $f_{21}$  receives a token. Consequently, the message *Fire fighter blocked* is sent to the fire control system that will ask for human intervention. Notably, to better explain the described scenarios, we provide a video [1] that shows the entire execution of the running example employing the animation of tokens over the BPMN model and the place graph.

Summing up, this example shows how a simple interaction with the environment, like closing a door, can impact the outcome of a collaboration. This paves the way for reasoning on different scenarios where the physical environment can influence the process execution and vice-versa.

## 5 Related work

Concerning the literature that discusses the interplay between BPMN and the physical environment, Decker et al. [4] propose a modeling approach that assigns location conditions to BPMN tasks so that their execution is constrained by spatial aspects.

Dorndorfer et al. [5] propose an extension to BPMN notation where spatial information can trigger conditional boundary events attached to tasks to handle exceptional behaviors. Similarly, Zhu et al. [20] extend the BPMN notation by defining a new type of task called location-dependent task. The execution of this kind of task is restricted to desired spatial locations. Also, they define location-dependent exclusive gateways which base their decisions on environmental conditions. Tomas Kozel [8] does a step forward proposing a graphical extension to BPMN that focuses on mobility. This work gives a categorization of mobile objects and introduces location-based events to trigger process activities based on changes in the physical location. Poss et al. [11] introduce in the BPMN meta-model novel concepts related to the environment such as location data, location-based resource allocation, and location events. Nevertheless, these works focus only on BPMN processes, adding constraints and handlers based on environmental aspects. Moreover, they do not provide any spatial model and do not keep track of the participant's position during the process execution.

An approach that overcomes this limitation is the one of Saddem-Yagoubi et al. [15], which define a formal temporal logic that extends BPMN elements, such as gateways and conditional sequence flows, to incorporate location information and enable location-dependent decision-making. Concerning the spatial model, they describe the space graphs only formally, without a graphical representation. Moreover, they provide a model checker for the verification of behavioral properties on the produced models. Nevertheless, they focus mostly on the effect of the environment on process execution while leaving under specified the influence of process activities on the environment, both in terms of topology and spatial attributes.

Summing up, even though the literature lately is increasingly dealing with the concept of the environment within BPMN, our approach advances the current state of the art for different aspects, as resumed in Table 1. To the best of our knowledge, ours is the only formal approach that allows altogether to: keep track of participants' position during the process execution, model the physical environment and its attributes explicitly, link the task execution and the environment by means of environmental constraints (guards) and modifications (assignments), and model tasks that involve a movement in the space. Moreover, our approach allows the definition of environmental conditions on decision gateways and on events, both boundary and intermediate, all this without introducing new BPMN elements.

## 6 Concluding remarks

This work discusses the interplay between BPMN collaborations and the physical environment. In particular, we prescribe the use of place graphs and place attributes as an abstraction of the physical environment where the collaborations take place. We link these concepts representing the physical environment with the BPMN meta-model resulting in the definition of environmental BPMN collaborations. Finally, we provide

Paper	Formal Semantics	Spatial Model		Task			Location -based Gateway	Location -based Events	New BPMN Elements	Participant Tracking
		Topology	Place attributes	Env. constraint	Env. modification	Movement				
[4]	no	no	no	yes	no	no	no	no	no	no
[5]	no	no	no	yes	no	no	no	no	no	no
[20]	no	no	no	yes	no	no	yes	no	no	no
[8]	no	no	no	yes	no	yes	no	yes	yes	no
[11]	no	no	no	no	no	no	no	yes	yes	no
[15]	yes	yes	no	yes	no	yes	yes	no	no	yes
our	yes	yes	yes	yes	yes	yes	yes	yes	no	yes

Table 1: Resuming related work Table

a formal operational semantics of the environmental BPMN collaboration models and illustrate and assess it on a running example.

**Discussion.** We merge business processes in the form of BPMN collaborations and the environment in the form of place graphs and place attributes. The choice of adopting the BPMN notation depends on its large application [19]. In particular, we use collaboration diagrams to model of process aspects of interest when dealing with the environment. Reasoning on multiple participants amplifies the interaction with the environment, while communication enables the sharing of environmental information. At the same time, we intentionally left out of the discussion those BPMN elements that draw away the reader from the addressed problem, e.g., multiple instances or sub-processes. Notably, we decided to extend the BPMN meta-model with environmental concepts without modifying the graphical notation so that designers are not required to learn new elements, and existing modeling tools can be used without customization. Nevertheless, graphical enhancements of the model that show the link between the BPMN model and the place graph can facilitate comprehension. This objective could be achieved with the support of custom BPMN modeling or animation tools.

Concerning the environment, we use place graph models to give an abstract representation of the space rather than using geometric models which represent the space with faithful measures. In particular, place graphs represent space at varying levels of abstraction tailored to specific domains; for instance, a place may correspond to a room or a tile. This flexibility enables the modeling of varying degrees of spatial detail. Anyway, our approach assumes that participants of the environmental BPMN collaboration possess a priori knowledge regarding the environment where they can move. Furthermore, we consider some aspects of the environment as static, e.g., it forbids the addition or removal of places, as well as the insertion of obstacles that may impede (even partially) the access to specific places. Another assumption we made on the place graphs is that places are equidistant with each other and no weight function is defined on edges for representing such environmental concepts as distance and gradient.

Nevertheless, place graphs can be extended to support these omitted features by adding to the semantics new spatial assignments that act on the places. Moreover, they could support weight functions for individual edges to accommodate diverse and more flexible spatial scenarios.

**Future Work.** In future work, we intend to continue the investigation of the interplay between BPMN collaborations and the environment enriching the spatial model as well



as the formal account as mentioned above. Moreover, we plan to work on animation and verification techniques to ease the understanding of our environmental BPMN models and enable formal reasoning.

## References

1. Companion technical report (2024), available at: <https://anonymous.4open.science/r/BPM2024>
2. Afyouni, I., Ray, C., Christophe, C.: Spatial models for context-aware indoor navigation systems: A survey. *Journal of Spatial Information Science* **1**(4), 85–123 (2012)
3. Aoumeur, N., Fiadeiro, J., Oliveira, C.: Towards an architectural approach to location-aware business process. In: *Enabling Technologies: Infrastructure for Collaborative Enterprises*. vol. 13, pp. 147–152. IEEE (2004)
4. Decker, M., Che, H., Oberweis, A., Stürzel, P., Vogel, M.: Modeling mobile workflows with bpmn. In: *Mobile Business and Mobility Roundtable*. pp. 272–279. IEEE (2010)
5. Dörndorfer, J., Seel, C.: A framework to model and implement mobile context-aware business applications. In: *Modellierung 2018*, pp. 23–38. Gesellschaft für Informatik e.V. (2018)
6. Hermann, A., Scholta, H., Bräuer, S., Becker, J.: Collaborative business process management - a literature-based analysis of methods for supporting model understandability. In: *Towards Thought Leadership in Digital Transformation*. *Wirtschaftsinformatik* (2017)
7. Jensen, C., Lu, H., Yang, B.: Graph model based indoor tracking. In: *Mobile Data Management: Systems, Services and Middleware*. pp. 122–131. ACM (2009)
8. Kozel, T.: Bpmn mobilisation. In: *Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable*. p. 307–310. World Scientific and Engineering Academy and Society (2010)
9. Monsalve, C., April, A., Abran, A.: Business process modeling with levels of abstraction. In: *Conference on Communication and Computing*, pp. 13–15. IEEE (2015)
10. OMG: Business process model and notation. (BPMN V2.0) (2011)
11. Poss, L., Dietz, L., Schönig, S.: Labpmn: Location-aware business process modeling and notation. In: *Cooperative Information Systems*. vol. 14353, pp. 198–216. Springer (2024)
12. Poss, L., Schönig, S.: A generic approach towards location-aware business process execution. In: *Enterprise, Business-Process and Information Systems Modeling*. LNCS, vol. 479, pp. 103–118. Springer (2023)
13. Rosemann, M., Recker, J.: Context-aware process design: Exploring the extrinsic drivers for process flexibility. In: *Proceedings of the Workshops and Doctoral Consortium*, pp. 149–158. Namur University Press (2006)
14. Rosemann, M., Recker, J., Flender, C.: Contextualisation of business processes. *International Journal of Business Process Integration and Management* **3**(1), 47–60 (2008)
15. Saddem-Yagoubi, R., Poizat, P., Houhou, S.: Business processes meet spatial concerns: the sbpmn verification framework. In: *Formal Methods*. pp. 218–234. Springer (2021)
16. Stropi, L.J.R., Chiotti, O., Villarreal, P.D.: Extending BPMN 2.0: Method and Tool Support. In: *Business Process Model and Notation*, pp. 59–73. Springer (2011)
17. Surname, N., Surname, N., Surname, N., Surname, N., Surname, N.: Title. *Journal Volume*(Number), Pages (Year)
18. Surname, N., Surname, N., Surname, N., Surname, N., Surname, N.: Title. *Journal Volume*, Pages (Year)
19. Surname, N., Surname, N., Surname, N., Surname, N., Surname, N.: Title. In: *Conference paper*, p. Pages. Publisher (Year)
20. Zhu, X., Recker, J., Zhu, G., Maria Santoro, F.: Exploring location-dependency in process modeling. *Business Process Management Journal* (2014)

## **A One-to-one correspondence between the BPMN graphical notation, the textual notation and the BPMN XML format.**

In this appendix, we show the one-to-one correspondence between the BPMN graphical notation, the textual notation, and the BPMN XML format. Table 2 regards pool and events, Table 3 regards gateways, and Table 4 regards tasks.

## B Additional example of a table service in a restaurant

In this appendix, we provide a further example to show the approach's applicability in another domain. Figure 9 depicts the BPMN collaboration model about a table service in a restaurant.

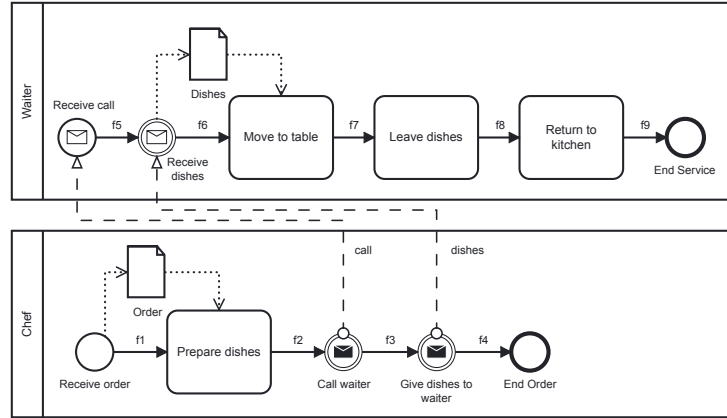


Fig. 9: Table service collaboration model

The collaboration is performed by a *waiter*, and a *chef* which collaborate to prepare and leave dishes to their customers in a restaurant. The collaboration starts when the chef receives the order. As soon as the chef gets it, he/she prepares the dishes. Once ready, the chef calls the waiter and gives him/her the ordered dishes. The waiter moves to the table, leaves the dishes to the customers sitting there, and finally returns to the kitchen ending its service. Concerning the environment, the model of the restaurant comprises a kitchen, a dining area, and a bathroom. Figures 10,11 and 12 represent three versions, namely case 1, case 2, and case 3, of the space model depending on the tables' position and the presence of obstacles to show different outcomes of the collaboration.

The union of the described BPMN collaboration model and the restaurant represented as a place graph, results in the environmental BPMN collaboration model of the table service. Its structure is  $C_r = \text{pool}(p_c, P_c) \mid \text{pool}(p_w, P_w)$ , and the space models for the three cases are respectively  $S_r^1$ ,  $S_r^2$  and  $S_r^3$ , see Figure 13 for a complete account. The initial configuration of the environmental BPMN collaboration model is the following:  $\langle C_r, S_r, \sigma_f, \sigma_a^0, \sigma_t^0, \sigma_p, \sigma_m^0 \rangle$ , where  $\sigma_f = \sigma_f^0 \cdot [f'_1 \mapsto 1]$  and  $\sigma_p = \sigma_p^0 \cdot [p_w \mapsto pl_7]$ .

Note that in each case we assume that the waiter has to move from the kitchen (position  $pl_7$ ) to serve the guest sitting in the bottom-left of the dining area from  $pl_{25}$ . Therefore, the three cases differ in the execution of task *Move to table* performed by the waiter by means of rule  $P\text{-Task}_M$ .

- In the **case 1**, the waiter can choose to follow the shortest path from  $pl_7$  to the destination place  $pl_{25}$  for instance, he/she can pass these locations in 8 steps:  $(pl_7, pl_6, pl_5, pl_4, pl_{11}, pl_{19}, pl_{18}, pl_{17}, pl_{25})$ .

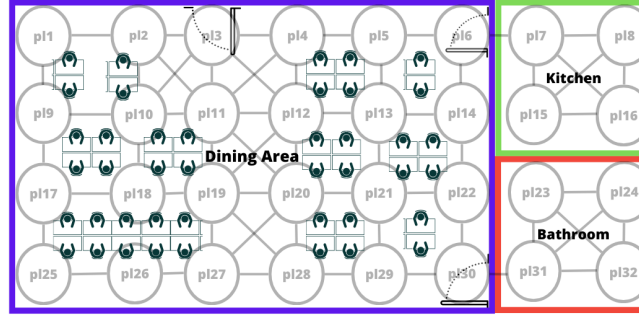


Fig. 10: Floor plan and place graph of a restaurant case 1

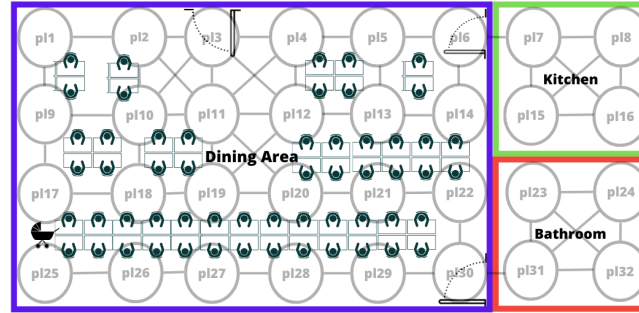


Fig. 11: Floor plan and place graph of a restaurant case 2

- In the **case 2**, the waiter applies the same strategy. However, the table disposition is changed and there is a stroller that blocks the link between  $pl_{17}$  and  $pl_{25}$ . Thus, the waiter have to pass from the following locations in 14 steps: ( $pl_7, pl_6, pl_5, pl_4, pl_{11}, pl_{19}, pl_{20}, pl_{21}, pl_{22}, pl_{30}, pl_{29}, pl_{28}, pl_{27}, pl_{26}, pl_{25}$ ) taking longer distance and time respect to the Case 1 (Figure 11).
- In the **case 3**, the composition of the restaurant is changed again, indeed, the stroller and the table disposition block the passage to reach the place  $pl_{25}$ . In this situation, the condition of rule  $P\text{-}Task_M$  is not satisfied, indeed,  $next(S, \sigma_p, p, D) = \emptyset$ . Therefore, the task *Move to table* is in a deadlock due to a spatial concern.

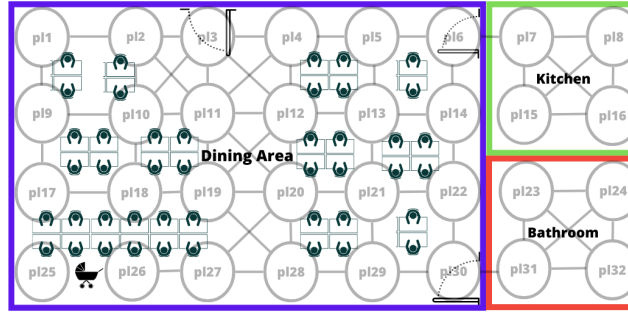


Fig. 12: Floor plan and place graph of a restaurant case 3

$C_d = \text{pool}(p_c, P_c) \mid \text{pool}(p_w, P_w)$
$P_c = \text{start}(f'_1, f_1) \mid \text{task}(f_1, \text{Prepare dishes, true, Order.pos} := \text{pl}_{25}, \epsilon, \text{nil, nil}, f_2) \mid$ $\text{interSnd}(f_2, \text{call} : \text{Order.pos}, f_3) \mid \text{interSnd}(f_3, \text{dishes} : \text{Order.dishes}, f_3) \mid \text{end}(f_4)$
$P_w = \text{startRcv}(\text{call} : \text{Dishes.pos}, f_5) \mid \text{interRcv}(f_5, \text{dishes} : \text{Dishes.dishes}, f_6)$ $\text{task}(f_6, \text{Move to table, true, } \epsilon, \epsilon, \text{Dishes.pos, nil}, f_7) \mid \text{task}(f_7, \text{Leave dishes, true, } \epsilon, \epsilon, \text{nil, nil}, f_8) \mid$ $\text{task}(f_8, \text{Return to kitchen, true, } \epsilon, \epsilon, \text{pl}_{25}, \text{nil}, f_9) \mid \text{end}(f_9)$
$S_r^1 = (P_r, E^1) \quad P_r = \{p_j \mid 1 \leq j \leq 32\} \quad E^1 = \{(p_1, p_2), (p_1, p_9), (p_2, p_1), (p_2, p_3), (p_2, p_{10}), (p_2, p_{11}),$ $(p_3, p_2), (p_3, p_4), (p_3, p_{10}), (p_3, p_{11}), (p_3, p_{12}), (p_4, p_3), (p_4, p_5), (p_4, p_{11}), (p_4, p_{12}), (p_5, p_4), (p_5, p_6),$ $(p_5, p_{13}), (p_6, p_5), (p_6, p_7), (p_6, p_{14}), (p_7, p_6), (p_7, p_8), (p_7, p_{15}), (p_7, p_{16}), (p_8, p_7), (p_8, p_{15}), (p_8, p_{16}),$ $(p_9, p_1), (p_9, p_{10}), (p_9, p_{17}), (p_{10}, p_2), (p_{10}, p_3), (p_{10}, p_9), (p_{10}, p_{11}), (p_{10}, p_{18}), (p_{11}, p_2), (p_{11}, p_3), (p_{11}, p_4),$ $(p_{11}, p_{10}), (p_{11}, p_{12}), (p_{11}, p_{19}), (p_{11}, p_{20}), (p_{12}, p_3), (p_{12}, p_4), (p_{12}, p_{11}), (p_{12}, p_{13}), (p_{12}, p_{19}), (p_{12}, p_{20}),$ $(p_{13}, p_5), (p_{13}, p_{12}), (p_{13}, p_{14}), (p_{13}, p_{21}), (p_{14}, p_6), (p_{14}, p_{13}), (p_{14}, p_{22}), (p_{15}, p_7), (p_{15}, p_8), (p_{15}, p_{16}),$ $(p_{16}, p_7), (p_{16}, p_8), (p_{16}, p_{15}), (p_{17}, p_9), (p_{17}, p_{18}), (p_{17}, p_{25}), (p_{18}, p_{10}), (p_{18}, p_{17}), (p_{18}, p_{19}), (p_{19}, p_{11}),$ $(p_{19}, p_{12}), (p_{19}, p_{18}), (p_{19}, p_{20}), (p_{19}, p_{27}), (p_{19}, p_{28}), (p_{20}, p_{11}), (p_{20}, p_{12}), (p_{20}, p_{19}), (p_{20}, p_{21}), (p_{20}, p_{27}),$ $(p_{20}, p_{28}), (p_{21}, p_{13}), (p_{21}, p_{20}), (p_{21}, p_{22}), (p_{21}, p_{29}), (p_{22}, p_{14}), (p_{22}, p_{21}), (p_{22}, p_{30}), (p_{23}, p_{24}), (p_{23}, p_{31}),$ $(p_{23}, p_{32}), (p_{24}, p_{23}), (p_{24}, p_{31}), (p_{24}, p_{32}), (p_{25}, p_{17}), (p_{25}, p_{26}), (p_{26}, p_{25}), (p_{26}, p_{27}), (p_{27}, p_{19}), (p_{27}, p_{20}),$ $(p_{27}, p_{26}), (p_{27}, p_{28}), (p_{28}, p_{19}), (p_{28}, p_{20}), (p_{28}, p_{27}), (p_{28}, p_{29}), (p_{29}, p_{21}), (p_{29}, p_{28}), (p_{29}, p_{30}), (p_{30}, p_{22}),$ $(p_{30}, p_{29}), (p_{30}, p_{31}), (p_{31}, p_{23}), (p_{31}, p_{24}), (p_{31}, p_{30}), (p_{31}, p_{32}), (p_{32}, p_{23}), (p_{32}, p_{24}), (p_{32}, p_{31})\}$ $S_r^2 = (P_r, E^2) \quad P_r = \{p_j \mid 1 \leq j \leq 32\} \quad E^2 = \{(p_1, p_2), (p_1, p_9), (p_2, p_1), (p_2, p_3), (p_2, p_{10}), (p_2, p_{11}),$ $(p_3, p_2), (p_3, p_4), (p_3, p_{10}), (p_3, p_{11}), (p_3, p_{12}), (p_4, p_3), (p_4, p_5), (p_4, p_{11}), (p_4, p_{12}), (p_5, p_4), (p_5, p_6),$ $(p_5, p_{13}), (p_6, p_5), (p_6, p_7), (p_6, p_{14}), (p_7, p_6), (p_7, p_8), (p_7, p_{15}), (p_7, p_{16}), (p_8, p_7), (p_8, p_{15}), (p_8, p_{16}),$ $(p_9, p_1), (p_9, p_{10}), (p_9, p_{17}), (p_{10}, p_2), (p_{10}, p_3), (p_{10}, p_9), (p_{10}, p_{11}), (p_{10}, p_{18}), (p_{11}, p_2), (p_{11}, p_3), (p_{11}, p_4),$ $(p_{11}, p_{10}), (p_{11}, p_{12}), (p_{11}, p_{19}), (p_{11}, p_{20}), (p_{12}, p_3), (p_{12}, p_4), (p_{12}, p_{11}), (p_{12}, p_{13}), (p_{12}, p_{19}), (p_{12}, p_{20}),$ $(p_{13}, p_{12}), (p_{13}, p_{14}), (p_{14}, p_6), (p_{14}, p_{13}), (p_{15}, p_7), (p_{15}, p_8), (p_{15}, p_{16}), (p_{16}, p_7), (p_{16}, p_8), (p_{16}, p_{15}),$ $(p_{17}, p_9), (p_{17}, p_{18}), (p_{18}, p_{10}), (p_{18}, p_{17}), (p_{18}, p_{19}), (p_{19}, p_{11}), (p_{19}, p_{12}), (p_{19}, p_{18}), (p_{19}, p_{20}), (p_{20}, p_{11}),$ $(p_{20}, p_{19}), (p_{20}, p_{21}), (p_{21}, p_{20}), (p_{21}, p_{22}), (p_{22}, p_{21}), (p_{22}, p_{30}), (p_{23}, p_{24}), (p_{23}, p_{31}),$ $(p_{23}, p_{32}), (p_{24}, p_{23}), (p_{24}, p_{31}), (p_{24}, p_{32}), (p_{25}, p_{26}), (p_{26}, p_{25}), (p_{26}, p_{27}), (p_{27}, p_{19}), (p_{27}, p_{20}),$ $(p_{27}, p_{26}), (p_{27}, p_{28}), (p_{28}, p_{19}), (p_{28}, p_{20}), (p_{28}, p_{27}), (p_{28}, p_{29}), (p_{29}, p_{28}), (p_{29}, p_{30}), (p_{30}, p_{22}),$ $(p_{30}, p_{29}), (p_{30}, p_{31}), (p_{31}, p_{23}), (p_{31}, p_{24}), (p_{31}, p_{30}), (p_{31}, p_{32}), (p_{32}, p_{23}), (p_{32}, p_{24}), (p_{32}, p_{31})\}$ $S_r^3 = (P_r, E^3) \quad P_r = \{p_j \mid 1 \leq j \leq 32\} \quad E^3 = \{(p_1, p_2), (p_1, p_9), (p_2, p_1), (p_2, p_3), (p_2, p_{10}), (p_2, p_{11}),$ $(p_3, p_2), (p_3, p_4), (p_3, p_{10}), (p_3, p_{11}), (p_3, p_{12}), (p_4, p_3), (p_4, p_5), (p_4, p_{11}), (p_4, p_{12}), (p_5, p_4), (p_5, p_6),$ $(p_5, p_{13}), (p_6, p_5), (p_6, p_7), (p_6, p_{14}), (p_7, p_6), (p_7, p_8), (p_7, p_{15}), (p_7, p_{16}), (p_8, p_7), (p_8, p_{15}), (p_8, p_{16}),$ $(p_9, p_1), (p_9, p_{10}), (p_9, p_{17}), (p_{10}, p_2), (p_{10}, p_3), (p_{10}, p_9), (p_{10}, p_{11}), (p_{10}, p_{18}), (p_{11}, p_2), (p_{11}, p_3), (p_{11}, p_4),$ $(p_{11}, p_{10}), (p_{11}, p_{12}), (p_{11}, p_{19}), (p_{11}, p_{20}), (p_{12}, p_3), (p_{12}, p_4), (p_{12}, p_{11}), (p_{12}, p_{13}), (p_{12}, p_{19}), (p_{12}, p_{20}),$ $(p_{13}, p_{12}), (p_{13}, p_{14}), (p_{14}, p_6), (p_{14}, p_{13}), (p_{15}, p_7), (p_{15}, p_8), (p_{15}, p_{16}), (p_{16}, p_7), (p_{16}, p_8), (p_{16}, p_{15}),$ $(p_{17}, p_9), (p_{17}, p_{18}), (p_{18}, p_{10}), (p_{18}, p_{17}), (p_{18}, p_{19}), (p_{19}, p_{11}), (p_{19}, p_{12}), (p_{19}, p_{18}), (p_{19}, p_{20}), (p_{20}, p_{11}),$ $(p_{20}, p_{19}), (p_{20}, p_{21}), (p_{21}, p_{20}), (p_{21}, p_{22}), (p_{22}, p_{21}), (p_{22}, p_{30}), (p_{23}, p_{24}), (p_{23}, p_{31}),$ $(p_{23}, p_{32}), (p_{24}, p_{23}), (p_{24}, p_{31}), (p_{24}, p_{32}), (p_{25}, p_{26}), (p_{26}, p_{25}), (p_{26}, p_{27}), (p_{27}, p_{19}), (p_{27}, p_{20}),$ $(p_{27}, p_{26}), (p_{27}, p_{28}), (p_{28}, p_{19}), (p_{28}, p_{20}), (p_{28}, p_{27}), (p_{28}, p_{29}), (p_{29}, p_{28}), (p_{29}, p_{30}), (p_{30}, p_{22}),$ $(p_{30}, p_{29}), (p_{30}, p_{31}), (p_{31}, p_{23}), (p_{31}, p_{24}), (p_{31}, p_{30}), (p_{31}, p_{32}), (p_{32}, p_{23}), (p_{32}, p_{24}), (p_{32}, p_{31})\}$

Fig. 13: Textual representation of the table service collaboration.

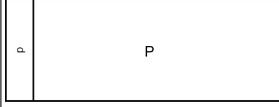

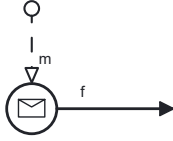

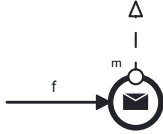
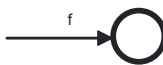
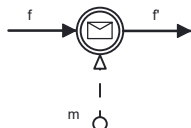
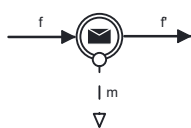

BPMN elements	Textual Notation	BPMN XML syntax
	$\text{pool}(p, P)$	<code>&lt;participant name="p" ... /&gt;</code>
	$\text{start}(f, f')$	<code>&lt;startEvent ... &gt;   &lt;outgoing&gt;f&lt;/outgoing&gt; &lt;/startEvent&gt;</code>
	$\text{startRcv}(m:\text{do.a}, f)$	<code>&lt;startEvent ... &gt;   &lt;outgoing&gt;f&lt;/outgoing&gt;   &lt;messageEventDefinition ... /&gt;   &lt;bpmn:extensionElements&gt;     &lt;bpmnenv:field field="do.a" /&gt;   &lt;/bpmn:extensionElements&gt; &lt;/startEvent&gt;</code>
	$\text{startCond}(\text{exp}, f)$	<code>&lt;startEvent .... &gt;   &lt;outgoing&gt;f&lt;/outgoing&gt;   &lt;conditionalEventDefinition ... &gt;     &lt;condition ... &gt;exp&lt;/condition&gt;   &lt;/conditionalEventDefinition&gt; &lt;/startEvent&gt;</code>
	$\text{endSnd}(f, m:\text{exp})$	<code>&lt;endEvent ... &gt;   &lt;incoming&gt;f&lt;/incoming&gt;   &lt;messageEventDefinition ... /&gt;   &lt;bpmn:extensionElements&gt;     &lt;bpmnenv:field field="exp" /&gt;   &lt;/bpmn:extensionElements&gt; &lt;/endEvent&gt;</code>
	$\text{end}(f)$	<code>&lt;endEvent ... &gt;   &lt;incoming&gt;f&lt;/incoming&gt; &lt;/endEvent&gt;</code>
	$\text{interRcv}(f, m:\text{do.a}, f')$	<code>&lt;intermediateCatchEvent ... &gt;   &lt;incoming&gt;f&lt;/incoming&gt;   &lt;outgoing&gt;f'&lt;/outgoing&gt;   &lt;messageEventDefinition ... /&gt;   &lt;bpmn:extensionElements&gt;     &lt;bpmnenv:field field="do.a" /&gt;   &lt;/bpmn:extensionElements&gt; &lt;/intermediateCatchEvent&gt;</code>
	$\text{interSnd}(f, m:\text{exp}, f')$	<code>&lt;intermediateThrowEvent ... &gt;   &lt;incoming&gt;f&lt;/incoming&gt;   &lt;outgoing&gt;f'&lt;/outgoing&gt;   &lt;messageEventDefinition ... /&gt;   &lt;bpmn:extensionElements&gt;     &lt;bpmnenv:field field="exp" /&gt;   &lt;/bpmn:extensionElements&gt; &lt;/intermediateThrowEvent&gt;</code>
	$\text{interCondCatch}(f, \text{exp}, f')$	<code>&lt;intermediateCatchEvent ... &gt;   &lt;incoming&gt;f&lt;/incoming&gt;   &lt;outgoing&gt;f'&lt;/outgoing&gt;   &lt;conditionalEventDefinition .... &gt;     &lt;condition ... &gt;exp&lt;/condition&gt;   &lt;/conditionalEventDefinition&gt; &lt;/intermediateCatchEvent&gt;</code>

Table 2: Correspondence between graphical, textual and BPMN XML notation: pool and events.

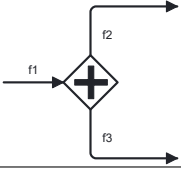
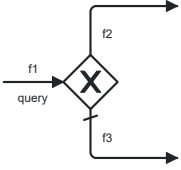
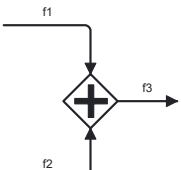
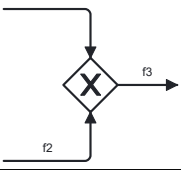
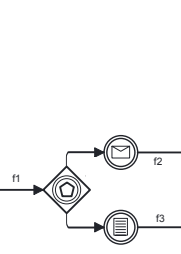
Graphical Notation	Textual Notation	BPMN XML syntax
	<code>andSplit(f1, {f2, f3})</code>	<pre>&lt;parallelGateway ... &gt;   &lt;incoming&gt;f1 &lt;/incoming&gt;   &lt;outgoing&gt;f2 &lt;/outgoing&gt;   &lt;outgoing&gt;f3 &lt;/outgoing&gt; &lt;/parallelGateway &gt;</pre>
	<code>xorSplit(f1, {(f2, do.a &gt; 0), (f3, default)})</code>	<pre>&lt;exclusiveGateway ... default="f3" &gt;   &lt;incoming&gt;f1 &lt;/incoming&gt;   &lt;outgoing&gt;f2 &lt;/outgoing&gt;   &lt;outgoing&gt;f3 &lt;/outgoing&gt; &lt;/exclusiveGateway &gt; ... &lt;sequenceFlow id="f1"&gt;   &lt;conditionExpression ... &gt;do.a     &gt; 0&lt;/conditionExpression   &gt; &lt;/sequenceFlow &gt;</pre>
	<code>andJoin({f1, f2}, f3)</code>	<pre>&lt;parallelGateway ... &gt;   &lt;incoming&gt;f1 &lt;/incoming&gt;   &lt;incoming&gt;f2 &lt;/incoming&gt;   &lt;outgoing&gt;f3 &lt;/outgoing&gt; &lt;/parallelGateway &gt;</pre>
	<code>xorJoin({f1, f2}, f3)</code>	<pre>&lt;exclusiveGateway ... &gt;   &lt;incoming&gt;f1 &lt;/incoming&gt;   &lt;incoming&gt;f2 &lt;/incoming&gt;   &lt;outgoing&gt;f3 &lt;/outgoing&gt; &lt;/exclusiveGateway &gt;</pre>
	<code>eventBased(f1, (m2:do.a, f2), (exp, f3))</code>	<pre>&lt;eventBasedGateway ... &gt;   &lt;incoming&gt;f1 &lt;/incoming&gt;   ... &lt;/eventBasedGateway &gt; &lt;intermediateCatchEvent ... &gt;   ...   &lt;outgoing&gt;f2 &lt;/outgoing&gt;   &lt;messageEventDefinition ... /&gt;   &lt;bpmn:extensionElements&gt;     &lt;bpmnenv:field field="do.a" /&gt;   &lt;/bpmn:extensionElements&gt; &lt;/intermediateCatchEvent &gt; &lt;intermediateCatchEvent ... &gt;   ...   &lt;outgoing&gt;f3 &lt;/outgoing&gt;   &lt;conditionalEventDefinition     .... &gt;     &lt;condition ... &gt;exp&lt;/       condition &gt;     &lt;/conditionalEventDefinition &gt; &lt;/intermediateCatchEvent &gt;</pre>

Table 3: Correspondence between graphical, textual and BPMN XML notation: gateways.


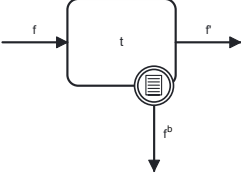
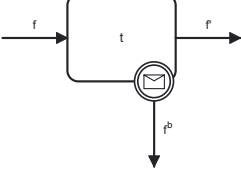
Graphical Notation	Textual Notation	BPMN XML syntax
	$\text{task}(f, t, \text{exp}, \text{do.a} = 1, \text{pl.a} = 0, \text{pl}, \text{nil}, f')$	<pre> &lt;task name="t" ...&gt;   &lt;incoming&gt;f&lt;/incoming&gt;   &lt;outgoing&gt;f'&lt;/outgoing&gt;   &lt;bpmn:extensionElements&gt;     &lt;bpmnenv:destination&gt;       pl     &lt;/bpmnenv:destination&gt;     &lt;bpmnenv:guard&gt;       exp     &lt;/bpmnenv:guard&gt;     &lt;bpmnenv:assignments&gt;       &lt;bpmnenv:assignment&gt;         do.a = 1       &lt;/bpmnenv:assignment&gt;       &lt;bpmnenv:assignment&gt;         pl.a = 0       &lt;/bpmnenv:assignment&gt;     &lt;/bpmnenv:assignments&gt;   &lt;/bpmn:extensionElements&gt; &lt;/task&gt; </pre>
	$\text{task}(f, t, \text{true}, \varepsilon, \varepsilon, \text{nil}, \text{boundCond}(\text{exp}, f^b), f')$	<pre> &lt;task name="t" ...&gt;   &lt;incoming&gt;f&lt;/incoming&gt;   &lt;outgoing&gt;f'&lt;/outgoing&gt; &lt;/task&gt; &lt;boundaryEvent ...&gt;   &lt;outgoing&gt;f^b&lt;/outgoing&gt;   &lt;conditionalEventDefinition ...&gt;     &lt;condition ...&gt;exp&lt;/condition&gt;   &lt;/conditionalEventDefinition&gt; &lt;/boundaryEvent&gt; </pre>
	$\text{task}(f, t, \text{true}, \varepsilon, \varepsilon, \text{nil}, \text{boundRcv}(m:\text{do.a}, f^b), f')$	<pre> &lt;task name="t" ...&gt;   &lt;incoming&gt;f&lt;/incoming&gt;   &lt;outgoing&gt;f'&lt;/outgoing&gt; &lt;/task&gt; &lt;boundaryEvent ...&gt;   &lt;outgoing&gt;f^b&lt;/outgoing&gt;   &lt;messageEventDefinition .../&gt;   &lt;bpmn:extensionElements&gt;     &lt;bpmnenv:field field="do.a" /&gt;   &lt;/bpmn:extensionElements&gt; &lt;/boundaryEvent&gt; </pre>

Table 4: Correspondence between graphical, textual and BPMN XML notation: tasks.