Companion Technical Report Associated to the Paper "Modeling, Formalizing, and Animating Environment-Aware BPMN Collaborations"

Abstract. In this technical report, we provide the full account of the formal semantics of the *environment-aware BPMN collaboration* models introduced in the paper "Modeling, Formalizing, and Animating Environment-Aware BPMN Collaborations", which is under submission.

1 Formal Syntax

In this section, we provide the formalization of the syntax of *environment-aware BPMN collaboration* models. Our syntax extends the one proposed in [1] with new BPMN elements, specifically devised to deal with the interaction between collaboration processes and the environment. Most of all, our formalization considers in addition a richer model of the environment. 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.

The formalization resorts to a textual representation that uses a Backus-Naur Form (BNF) syntax for representing the new BPMN elements and the extension of the environment. 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 textual notation makes explicit those technical details of collaboration models that are not part of the graphical representation (e.g., message payloads, assignments, guard conditions), but are part of the low-level XML characterization of the model. This information is needed to properly define the execution semantics of the environment-aware BPMN collaboration models.

Figure 1 reports the BNF syntax defining the textual notation describing the structure of environment-aware BPMN collaboration models. Specifically, the upper table reports the grammar productions defining the syntax of collaboration models together with the notation of the related generic elements of the syntactic categories. The symbol $*$ stands for (possibly empty) sequences, and $+$ for non-empty sequences. As usual for BPMN models, we assume that message flows, sequence flows, and task names are unique in the model; similarly, we also assume that handshake flows are unique. Notably, even if this syntax would allow writing collaboration models that cannot be expressed in BPMN, we will consider only those terms of the syntax that can be derived from BPMN models. The bottom table reports the definition of the environment where a collaboration is enacted, together with the related notation.

Intuitively, a environment-aware BPMN collaboration is the union of the terms C, indicating the structure of the collaboration, and ENV, indicating the environmental

$C ::=$		*(Collaboration Structures)*
	$\mathsf{pool}(\mathsf{p}, P) \mid C \parallel C'$	(pool, pool composition)
$P ::=$		*(Process Structures)*
	$\mathsf{start}(\mathsf{f}, \mathsf{f}') \mid \mathsf{end}(\mathsf{f})$	(start event, end event)
$\mid$	$\mathsf{snd}(\mathsf{f}, \mathsf{m}, \mathsf{exp}, \mathsf{f}')$	(msg sending intermediate event)
$\mid$	$\mathsf{rcv}(\mathsf{f}, \mathsf{m}, \mathsf{do.i}, \mathsf{f}')$	(msg receiving intermediate event)
$\mid$	$\mathsf{andSplit}(\mathsf{f}, F) \mid \mathsf{andJoin}(F, \mathsf{f})$	(AND split/join gateway)
$\mid$	$\mathsf{xorSplit}(\mathsf{f}, (\mathsf{exp}, \mathsf{f})^{+}) \mid \mathsf{xorJoin}(F, \mathsf{f})$	(XOR split/join gateway)
$\mid$	$\mathsf{eventBased}(\mathsf{f}, (\mathsf{m}, \mathsf{do.i}, \mathsf{f}')^{+})$	(event-based gateway)
$\mid$	$T \mid P \| P'$	(tasks and process composition)
$T ::=$	$\mathsf{task}(\mathsf{f}, \mathsf{n}, \mathsf{exp}, A, \mathsf{d}, \mathsf{f}')$	(basic task)
$\mid$	$\mathsf{taskM}(\mathsf{f}, \mathsf{n}, \mathsf{exp}, A, \mathsf{exp}', \mathsf{f}')$	(movement task)
$\mid$	$\mathsf{taskB}(\mathsf{f}, \mathsf{h}, \mathsf{f}')$	(binding task)
$\mid$	$\mathsf{taskU}(\mathsf{f}, \mathsf{h}, \mathsf{f}')$	(unbinding task)
$A ::=$	$\mathsf{assign}(\mathsf{exp}, v)^{*}$	*(Assignments)*

(Notation)

Pool name: p	Data object name: do
Sequence flow: f	Physical place name: pp
Sequence flow set: F	Logical place name: lp
Message flow: m	Edge name: e
Handshake flow: h	Field name: i
Expression: exp	Attribute name: a
Task name: n	Variable (i.e., $\mathsf{do.i}$, $\mathsf{pp.a}$, $\mathsf{lp.a}$, $\mathsf{e.a}$): v

$ENV ::=$	(PL, LL)		*(Environment Model)*
$PL ::=$	$(\mathbb{PP}, \mathbb{E}, E, \mathbb{A}_{pp}, \mathbb{A}_{e})$	with $E \subseteq \mathbb{PP} \times \mathbb{E} \times \mathbb{PP}$	*(Physical Layer)*
$LL ::=$	(V^{*}, LP)	with $LP : \mathbb{LP} \rightarrow \mathbb{EXP}$	*(Logical Layer)*
$V ::=$	$(\mathbb{LP}, \mathbb{A}_{lp}, A_{lp})$	with $A_{lp} : \mathbb{A}_{lp} \rightarrow (\mathbb{AG} \times \mathbb{DAG})$	*(View)*

(Notation)

Physical place names set: $\mathbb{PP}$	Logical place names set: $\mathbb{LP}$
Edge names set: $\mathbb{E}$	Logical place attributes: $\mathbb{A}_{lp}$
Edges: E	Attribute (dis)aggregation: A_{lp}
Physical place attributes: $\mathbb{A}_{pp}$	Expressions set: $\mathbb{EXP}$
Edge attributes: $\mathbb{A}_{e}$	Aggregation functions set: $\mathbb{AG}$
Logical place definition function: LP	Disaggregation functions set: $\mathbb{DAG}$

Fig. 1: BNF syntax of environment-aware BPMN collaboration models.

model. A collaboration is rendered in the presented syntax as a collection of pools coupled with an environmental model divided into a physical layer and a logical layer. Formally, a collaboration C is a composition by means of the operator $\|$, of pool elements $\mathsf{pool}(\mathsf{p}, P)$ 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 $\mathsf{sans\ serif}$ font) by means of the operator $|$. We identify collaborations (resp. processes) up to commutativity and associativity of pool (resp. process element) collection. Thus, e.g., $C_1 \parallel C_2$ and $C_2 \parallel C_1$ are identified as the same collaboration due to the commutativity property, while $P_1 \mid (P_2 \mid P_3)$ and $(P_1 \mid P_2) \mid P_3$ are identified as the same process due to the associativity property. To support a compositional approach, in the textual notation each sequence/message flow of 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/message flow name.

More specifically, the elements of the notation are as follows.

- $\mathsf{pool}(\mathsf{p}, P)$ denotes a pool identified by a name p, enveloping a process structure P.
- $\mathsf{start}(\mathsf{f}, \mathsf{f}')$ denotes a start event that contains an incoming (spurious) sequence flow f, named enabling flow, used to activate the start event, and the outgoing sequence flow f'.
- $\mathsf{end}(\mathsf{f})$ denotes the end event and f is the incoming sequence flow.
- $\mathsf{snd}(\mathsf{f}, \mathsf{m}, \mathsf{exp}, \mathsf{f}')$ denotes the message intermediate sending event that is characterized by an incoming sequence flow f, a message m, an expression exp, whose evaluation will return the value to be sent, and an outgoing sequence flow f'.
- $\mathsf{rcv}(\mathsf{f}, \mathsf{m}, \mathsf{do.i}, \mathsf{f}')$ denotes the message intermediate receiving event that is characterized by an incoming sequence flow f, a message m, a data object field $\mathsf{do.i}$, and an outgoing sequence flow f'.
- $\mathsf{andSplit}(\mathsf{f}, F)$ denotes an AND split gateway characterized by an incoming sequence flow f and a set of outgoing sequence flow F.
- $\mathsf{andJoin}(F, \mathsf{f})$ denotes an AND join gateway characterized by a set of incoming sequence flow F and an outgoing sequence flow f.
- $\mathsf{xorSplit}(\mathsf{f}, (\mathsf{exp}, \mathsf{f})^+)$ denotes an XOR split gateway characterized by an incoming sequence flow f and a list of outgoing sequence flows, each one specifying a triggering condition exp.
- $\mathsf{xorJoin}(F, \mathsf{f})$ denotes XOR join gateways characterized by a set of incoming sequence flow F and an outgoing sequence flow f.
- $\mathsf{eventBased}(\mathsf{f}, (\mathsf{m}, \mathsf{do.i})^+)$ denotes an event-based gateway which is similar to the XOR-split gateway, but its outgoing sequence flow activation depends on taking place of message receiving events.
- $\mathsf{task}(\mathsf{f}, \mathsf{n}, \mathsf{exp}, A, \mathsf{d}, \mathsf{f}')$ denotes a basic task, which has a name n, a guard exp that is a conditional expression predicating on attributes, a list of assignments A, and a duration d that represents the amount of time it takes to execute the task. The task has also an incoming and an outgoing sequence flow. Note that, for the sake of readability, when the guard and/or the assignments are not used, they are omitted from the syntactic definition of the element .
- $\mathsf{taskM}(\mathsf{f}, \mathsf{n}, \mathsf{exp}, A, \mathsf{exp}', \mathsf{f}')$ denotes a movement task, which is a type of task that may involve a movement in the environment; it is similar to the basic task but it has

an additional expression exp' that represents the destination to reach. In addition, a movement task does not have the term d because it is assumed that its duration is given by the time to reach the destination.

- $\mathsf{taskB}(\mathsf{f},\mathsf{h},\mathsf{f}')$ denotes a binding task, which specifies a handshake flow h that connects it to the binding task of another participant (which has the same h).
- $\mathsf{taskU}(\mathsf{f},\mathsf{h},\mathsf{f}')$ denotes an unbinding task, which specifies a handshake flow h that connects it to the unbinding task of another participant (which has the same h).
- $\mathsf{assign}(\mathsf{exp},v)$ denotes an assignment that assigns the value resulting from the evaluation of the expression exp to a variable, which can be a data object field $\mathsf{do.i}$, an attribute of a physical place $\mathsf{pp.a}$, an attribute of a logical place $\mathsf{lp.a}$, or an attribute of an edge $\mathsf{e.a}$.

Let us focus now on the syntax of the environment model. An environment ENV is a tuple (PL,LL) where PL represents the physical layer while LL represents the logical layer.

The physical layer PL consists of a place graphs, i.e. a graph whose nodes represent physical places (whose names are in the set $\mathbb{PP}$) and directed arcs represent edges E (whose names are in the set $\mathbb{E}$) connecting places. Both physical places and edges are equipped with contextual information represented in terms of attributes (whose names belong to the sets $\mathbb{A}_{pp}$ and $\mathbb{A}_{e}$, respectively).

The logical layer LL consists of a set of views and a function $LP : \mathbb{LP} \rightarrow \mathbb{EXP}$ that specifies which physical places form each logical place. Notably, the physical places forming a logical place are not statically listed in the model because the constituents of a logical place may change dynamically. Thus, the LP function maps each logical place to a boolean expression on physical place attributes, whose evaluation on a physical place determines if it belongs (at the time of such evaluation) to the corresponding logical place. Formally, the set of physical places forming a given logical place $\mathsf{lp} \in \mathbb{LP}$ is defined as follows: $\mathscr{P}(\mathsf{lp}) = \{\mathsf{pp} \in \mathbb{PP} \mid [LP(\mathsf{lp})]_{\mathsf{pp}}\}$, where function $[\cdot]_{\mathsf{pp}}$ returns an expression resulting from the instantiation of the input expression with the physical place name pp, i.e., $[\mathsf{exp}]_{\mathsf{pp}}$ replaces each attribute a in exp with $\mathsf{pp.a}$.

Finally, a view V groups a set of logical places $\mathbb{LP}$, defining how the values of their attributes (in the set $\mathbb{A}_{lp}$) result from the aggregation of the values of physical place attributes and, vice versa, propagate from the logical to the physical layer. Specifically, the handling of the logical place attributes of a view is defined by the function A_{lp} that maps each logical attribute to an aggregation and a disaggregation function. The aggregating function of a logical attribute is called during the evaluation of an expression when the logical attribute is encountered; the disaggregating function, instead, is called when an assignment, having the logical attribute as target, is processed. In practice, these functions provide a transparent mechanism to access and update the value of a logical attribute (like a C# property). This allows the modeler to simply deal with logical attributes as standard (i.e., physical) attributes in expressions and assignments, thus relieving him/her from the duty of explicitly handling aggregation and disaggregation of values.

2 Formal Semantics

To describe the semantics of environment-aware BPMN collaboration, we enrich the structural part of the model with the execution state, given by: the marking of sequence flows with tokens [2, p. 27]; the value of the data object fields, physical place attributes and edge attributes; the binding among participants; the status of tasks; the position of the participants in the place graph; the status of timers; and the payloads of the exchanged messages. These stateful descriptions are called *configurations*; we have three kinds of configurations: a *process configuration* has the form $\langle P, ENV, \mathsf{p}, \sigma\rangle$, a *collaboration configuration* has the form $\langle C, ENV, \sigma\rangle$, and an *open-world environment configuration* has the form $[C, ENV, \sigma]$, where σ is an execution state. The last configuration differs from the previous one because it allows the environment to evolve freely and independently from the collaboration execution. The environment, indeed, can change based on external factors, such as the traffic on a road.

As mentioned before, the execution state carries many information concerning the execution of the environment-aware BPMN collaboration. In fact, the state σ consists of the following list of *state functions*:

- $\sigma_f : \mathbb{F} \rightarrow \mathbb{N}$ 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; σ_f^0 denotes the state where all sequence flows are unmarked except for the spurious flows, each one marked by one token;
- $\sigma_{doi} : (\mathbb{DO} \times \mathbb{I}) \rightarrow \mathbb{V}$ is a *data object state function* (where $\mathbb{DO}$ is the set of data object names, $\mathbb{I}$ is the set of data object fields, and $\mathbb{V}$ is the set of values) assigning values (possibly null) to data object fields; σ_{doi}^0 denotes the state where all data object attributes are set to null except for those data objects (the so-called *data inputs*) whose initial value is already provided by the modeler;
- $\sigma_{ppa} : (\mathbb{PP} \times \mathbb{A}_{pp}) \rightarrow \mathbb{V}$ is a *physical place attribute state function* assigning values (possibly null) to physical place attributes; σ_{app}^0 denotes the state where all place attributes are set to null or the initial value provided by the modeler;
- $\sigma_{ea} : (\mathbb{PP} \times \mathbb{A}_e) \rightarrow \mathbb{V}$ is an *edge attribute state function* assigning values (possibly null) to edge attributes; σ_{ape}^0 denotes the state where all edge attributes are set to null or the initial value provided by the modeler;
- $\sigma_h : \mathbb{P} \rightarrow 2^{\mathbb{P}}$ is a *handshake state function* (where $\mathbb{P}$ is the set of pool names, each one denoting a collaboration participant) that, given a participant, returns the set of participants with which it is bound;
- $\sigma_s : \mathbb{T} \rightarrow \{enab, dis\}$ is a *task state function* (where $\mathbb{T}$ is the set of task names) used to keep track of the active tasks; the status of a task depends on its evolution from enabled ($enab$) to disabled (dis) states and vice versa; σ_s^0 denotes the state where all tasks are disabled;
- $\sigma_p : \mathbb{P} \rightarrow \mathbb{PP}$ is the *place state function* mapping each participant to a place; σ_p^0 denotes the state where all places are empty except for those containing the initial positions of particpants provided by the modeler;
- $\sigma_t : \mathbb{T} \rightarrow \mathbb{N}$ is a *timer state function* specifying for each task's timer the time units to wait until it expires; σ_t^0 denotes the state where all timers are set to undef;
- $\sigma_m : \mathbb{M} \rightarrow 2^{\mathbb{V}^n}$ is a *message state function* (where $\mathbb{M}$ is the set of message flow names) that assigns to each message flow name m an ordered multiset of values

representing the messages received along m; σ_m^0 denotes the state where all message multisets are empty.

We define the update of state functions only for σ_f, as the update of the other state functions are similarly defined. The state obtained by updating in σ_f the number of tokens of the sequence flow f to n, written as $\sigma_f \cdot [\mathsf{f} \mapsto \mathsf{n}]$, is defined as follows: $(\sigma_f \cdot [\mathsf{f} \mapsto \mathsf{n}])(\mathsf{f}')$ returns n if $\mathsf{f}' = \mathsf{f}$, otherwise it returns $\sigma_f(\mathsf{f}')$.

The operational semantics of environment-aware BPMN collbaorations is defined employing *labeled transition systems* (LTSs). The LTS of the process behavior is a triple $\langle \mathscr{P}, \mathscr{L}, \rightarrow \rangle$ where: $\mathscr{P}$ is a set of processes configurations; $\mathscr{L}$, ranged over by λ, is a set of labels; and $\rightarrow \subseteq \mathscr{P} \times \mathscr{L} \times \mathscr{P}$ is a transition relation. The labels used by the transition relations $\rightarrow$ are generated by the following grammar:

$$\begin{array}{rll}
\eta ::= & & \textit{(Handshake labels)} \\
& \rightarrow \mathsf{h} \leftarrow & \quad \text{(binding)} \\
| & \leftarrow \mathsf{h} \rightarrow & \quad \text{(unbinding)} \\[1ex]
\varphi ::= & & \textit{(Untimed labels)} \\
& \eta & \quad \text{(handshake)} \\
| & \tau & \quad \text{(silent actions)} \\[1ex]
\lambda ::= & & \textit{(Process labels)} \\
& \varphi & \quad \text{(untimed action)} \\
| & \checkmark & \quad \text{(timed action)}
\end{array}$$

where $\checkmark$ denotes the passing of a unit of time (due, e.g., to movements in the environment), and τ denotes a silent transition (due, e.g., token flow in the processes). As simplification, $\langle P, ENV, \sigma \rangle \xrightarrow{\lambda} \langle P, ENV, \sigma' \rangle$ indicates that $(\langle P, ENV, \sigma \rangle, \lambda, \langle P, ENV, \sigma' \rangle) \in \rightarrow$, and says that the process P and the environment ENV in the state σ can do a transition labeled by λ and evolve to the state σ'. The LTS describing the behavior of collaborations (resp. open-world open-world environments) is a triple $\langle \mathscr{C}, \mathscr{L}', \rightarrow_c \rangle$ (resp. $\langle \mathscr{O}, \mathscr{L}', \rightarrow_o \rangle$) where: $\mathscr{C}$ (resp. $\mathscr{O}$) is a set of collaboration (resp. open-world environment) configurations, $\mathscr{L}'$, ranged over by α, is a set of labels; and $\rightarrow_c \subseteq \mathscr{C} \times \mathscr{L}' \times \mathscr{C}$ (resp. $\rightarrow_o \subseteq \mathscr{O} \times \mathscr{L}' \times \mathscr{O}$) is a transition relation. The labels used by these transition relations are generated by the follwoing grammar:

$$\begin{array}{rll}
\alpha ::= & & \textit{(Collaboration labels)} \\
& \tau & \quad \text{(silent action)} \\
| & \checkmark & \quad \text{(timed action)}
\end{array}$$

To improve the readability, the following simplifications reduce the notation of operational rules. We omit: *(i)* the environment ENV from the source configuration of transitions, as it is the same in all rules; *(ii)* the execution state σ from the source configuration of transitions, except in those rules with more than one transition in the premise; *(iii)* the process/collaboration structure and the environment from the target configuration of transitions, since they are not not affected by transitions; *(iv)* those state functions from target configurations that are not affected by transitions. Thus, for example, a transition $\langle P, ENV, \sigma_f, \sigma_{doi}, \sigma_{ppa}, \sigma_{ea}, \sigma_s, \sigma_p, \sigma_t, \sigma_m \rangle \xrightarrow{\lambda}$

$\langle P, ENV, \sigma'_f, \sigma'_{doi}, \sigma'_{ppa}, \sigma'_{ea}, \sigma'_s, \sigma'_p, \sigma'_t, \sigma'_m \rangle$ will be written as $P \xrightarrow{\lambda} \langle \sigma'_f \rangle$ when it simply affects the sequence flow state function.

Function $inc(\sigma_f, \mathsf{f}) = \sigma_f \cdot [\mathsf{f} \mapsto \sigma_f(\mathsf{f}) + 1]$, resp. $dec(\sigma_f, \mathsf{f}) = \sigma_f \cdot [\mathsf{f} \mapsto \sigma_f(\mathsf{f}) - 1]$, increments, resp. decrements, by one the number of tokens marking the sequence flow f in the state σ_f. They extend in a natural way to sets F of sequence flows; they are inductively defined as follows: $inc(\sigma_f, \emptyset) = dec(\sigma_f, \emptyset) = \sigma_f$, $inc(\sigma_f, \{\mathsf{f}\} \cup F) = inc(inc(\sigma_f, \mathsf{f}), F)$, $dec(\sigma_f, \{\mathsf{f}\} \cup F) = dec(dec(\sigma_f, \mathsf{f}), F)$. Function $dec(\sigma_t, \mathsf{n})$, which decrements the timer value for the timer n, is defined similarly.
Function $enab(\sigma_s, \mathsf{n}) = \sigma_s \cdot [\mathsf{n} \mapsto enab]$, resp. $dis(\sigma_s, \mathsf{n}) = \sigma_s \cdot [\mathsf{n} \mapsto dis]$, activates, resp. deactivates, the task n in the state σ_s.
Function $eval(ENV, \sigma_{doi}, \sigma_{ppa}, \sigma_{ea}, \sigma_p, \mathsf{exp}) = \mathsf{v}$ states that v is the value resulting from the evaluation of the expression exp on the data fields in σ_{doi}, attributes in σ_{ppa} and σ_{ea}, and participants' positions in σ_p. For logical place attributes, the evaluation resorts to the aggregation functions specified in the views of the environment model ENV. For the sake of presentation, we omit the environment and the state functions when they are clear from the context, thus writing $eval(\mathsf{exp})$.
Function $upd(ENV, \sigma_{doi}, \sigma_{ppa}, \sigma_{ea}, \sigma_p, A)$ performs the assignments A and returns the updated data object, physical place attribute and edge attributes state functions. For example, $upd(ENV, \sigma_{doi}, \sigma_{ppa}, \sigma_{ea}, \sigma_p, \mathsf{assign}(\mathsf{exp}, \mathsf{do.i}))$ = $(\sigma_{doi} \cdot [\mathsf{do.i} \mapsto eval(\mathsf{exp})], \sigma_{ppa}, \sigma_{ea})$. Differently, the assignment of a logical place attribute resorts to the disaggregation function specified in the environment model ENV. $upd(ENV, \sigma_{doi}, \sigma_{ppa}, \sigma_{ea}, \sigma_p, \mathsf{assign}(\mathsf{exp}, \mathsf{lp.a}))$ = $(\sigma_{doi}, \sigma'_{ppa}, \sigma_{ea})$ where σ'_{ppa} is returned by the call of the disaggreggation function associated to $\mathsf{lp.a}$. Formally, given the view $(\mathbb{LP}, \mathbb{A}_{lp}, A_{lp})$ in ENV such that $\mathsf{lp} \in \mathbb{LP}$, the disaggregation function corresponding to a is $A_{lp}(\mathsf{a}) \downarrow_2 = f_{dag}$ (where $\downarrow_i$ is a projection function that returns the second item of a pair). Thus, the call of disaggregation function is $f_{dag}(\mathsf{lp}, eval(\mathsf{exp}))$; we recall that aggregation/disaggregation functions are parametric with respect to the physical attribute on which they operate. For the sake of presentation, we omit the environment and the state functions when they are clear from the context, thus writing $upd(A)$.
Function $next(PL, \sigma_p, \mathsf{p}, pl)$, where pl can be either a logical place lp or a physical place pp, is defined as follows. If a physical place pp is given as input, i.e. $next(PL, \sigma_p, \mathsf{p}, \mathsf{pp})$, it returns the set of places that are next in the paths in the physical layer PL from the current place $\sigma_p(\mathsf{p})$ of the process participant to the destination pp; the function returns $\emptyset$ when there is no path to the destination. If a logical place lp is given as input, i.e. $next(PL, \sigma_p, \mathsf{p}, \mathsf{lp})$, it returns $next(PL, \sigma_p, \mathsf{p}, \mathsf{pp})$ where $\mathsf{pp} \in \mathscr{P}(\mathsf{lp})$. This means that, when the destination specified by the task is a logical place, the next place is computed, for the sake of simplicity, considering as destination one of the physical places (non-deterministically selected) forming the logical place. Notably, the *next* function determines the next place to reach by computing the shortest paths to the destination; different definitions can be provided for this function to specify different navigation strategies.
Function $arrived(\sigma_p, \mathsf{p}, pl)$, where pl can be either a logical place lp or a physical place pp, returns *true* if the current position of the participant p is the destination place pl. Formally, $arrived(\sigma_p, \mathsf{p}, \mathsf{pp}) = (\sigma_p(\mathsf{p}) = \mathsf{pp})$, and $arrived(\sigma_p, \mathsf{p}, \mathsf{lp}) = (\sigma_p(\mathsf{p}) \in \mathscr{P}(\mathsf{lp}))$.
Function $move(\sigma_p, \sigma_h, \mathsf{p}, \mathsf{pp})$ assigns to the the participant p, and all participants bound to it, the physical place pp in the state σ_p. Formally, $move(\sigma_p, \sigma_h, \mathsf{p}, \mathsf{pp}) = \sigma_p \cdot [\mathsf{p} \mapsto \mathsf{pp}] \cdot [\mathsf{p}_1 \mapsto \mathsf{pp}] \cdot \ldots \cdot [\mathsf{p}_n \mapsto \mathsf{pp}]$ with $\sigma_h(\mathsf{p}) = \{\mathsf{p}_1, \cdots, \mathsf{p}_n\}$.
Function $set(\sigma_t, \mathsf{n}, \mathsf{k}) = \sigma_t \cdot [\mathsf{n} \mapsto \mathsf{k}]$ sets the timer of the task named n to the natural number k.
Function $reset(\sigma_t, \mathsf{n}) = \sigma_t \cdot [\mathsf{n} \mapsto \mathsf{undef}]$ resets the timer of the task named n.
Function $add(\sigma_m, \mathsf{m}, \mathsf{v}) = \sigma_m \cdot [\mathsf{m} \mapsto \sigma_m(\mathsf{m}) \uplus \{\mathsf{v}\}]$ adds the value v for the message flow m in the state σ_m.
Function $rm(\sigma_m, \mathsf{m}, \mathsf{v}) = \sigma_m \cdot [\mathsf{m} \mapsto \sigma_m(\mathsf{m}) \setminus \{\mathsf{v}\}]$ removes the value v for message m from the state σ_m.
Function $getD(\mathsf{d})$ can be expressed in different ways. In this paper, duration d is considered as a couple (k_{min}, k_{max}) such that $getD(\mathsf{d})$ returns a uniformally distributed natural number in the interval $[k_{min}, k_{max}]$.
Function $notTimedEl(P)$ returns *false* when P is a basic task or a movement task, i.e., $notTimedEl(\mathsf{task}(\mathsf{f}, \mathsf{n}, \mathsf{exp}, A, \mathsf{d}, \mathsf{f}'))$ = $notTimedEl(\mathsf{taskM}(\mathsf{f}, \mathsf{n}, \mathsf{exp}, A, \mathsf{exp}', \mathsf{f}'))$ = *false*. For all other elements, this function returns *true*. Finally $notTimedEl(P_1 \mid P_2) = notTimedEl(P_1) \wedge notTimedEl(P_2)$.
Function $bind(\sigma_h, \mathsf{p}_1, \mathsf{p}_2)$ binds together the participants p_1 and p_2 in σ_h. Formally, $bind(\sigma_h, \mathsf{p}_1, \mathsf{p}_2) = \sigma'_h$ with $\sigma'_h(\mathsf{p}) = \sigma_h(\mathsf{p})$ if $\mathsf{p} \notin \{\mathsf{p}_1, \mathsf{p}_2\}$, $\sigma'_h(\mathsf{p}_1) = \sigma_h(\mathsf{p}_1) \cup \{\mathsf{p}_2\}$, and $\sigma'_h(\mathsf{p}_2) = \sigma_h(\mathsf{p}_2) \cup \{\mathsf{p}_1\}$. Notation $\sigma \cdot \sigma'_h$ denotes the update of state σ by replacing the enclosed σ_h by σ'_h.
Function $unbind(\sigma_h, \mathsf{p}_1, \mathsf{p}_2)$ unbinds the participants p_1 and p_2 in σ_h. Formally, $unbind(\sigma_h, \mathsf{p}_1, \mathsf{p}_2) = \sigma'_h$ with $\sigma'_h(\mathsf{p}) = \sigma_h(\mathsf{p})$ if $\mathsf{p} \notin \{\mathsf{p}_1, \mathsf{p}_2\}$, $\sigma'_h(\mathsf{p}_1) = \sigma_h(\mathsf{p}_1) \setminus \{\mathsf{p}_2\}$, and $\sigma'_h(\mathsf{p}_2) = \sigma_h(\mathsf{p}_2) \setminus \{\mathsf{p}_1\}$.
Function $evolve(\sigma)$ updates the environmental attributes in the state σ independently from the activities performed by the collaboration participants. The definition of this function is application dependent and must be provided by the modeller when the collaboration is executed in an open-world environment. By default, $evolve(\sigma) = \sigma$, meaning that the environment does not evolve independently if the modeler does not specify how.

Fig. 2: Auxiliary functions

Moreover, to further simplify the definition of the operational rules, in Figure 2 we define *auxiliary functions* that update the state functions of process configurations or performs checks in rules' conditions.

The operational rules defining the transition relations of the environment-aware BPMN semantics are given by the inference rules in Figures 3, 4, and 5.

We now briefly comment on the semantic rules, starting from those dealing with events and gateways in Figures 3. 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. 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-Rcv* is enabled when there is at least a token in the incoming flow and when there is a value in the message queue. As a result, this function increments the number of tokens in the outgoing sequence flow, updates the data object fields with the received value, and removes it from the queue. Rule *P-Snd* is enabled when there is at least a token in its incoming flow; as a result, the value resulting from the evaluation of the message expression exp is added to the queue of the message flow m, and the number of tokens in the outgoing sequence flow is incremented. Rule *P-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-XorSplit* 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-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. Rule *P-EventB* is activated when there is a token in the incoming flow and a value is present for at least one of the connected messages. The effect of the rule is similar to that of rule *P-Rcv*.

Let us consider the rules in Figure 4, dealing with basic tasks, movement tasks, binding and unbinding tasks. Rules $P\text{-}Task_{Act}$ and $P\text{-}TaskM_{Act}$ both activate tasks and movement tasks when they are disabled, there is a token in the incoming sequence flow, and the guard exp_g is satisfied. Then, they remove the token in the incoming sequence flow, enable the tasks, and start the timer of the task (not for the movement task). Rule $P\text{-}Task_{Elps1}$ decrements the value of the task's timer when it is enabled and the timeout is not expired. Rules $P\text{-}Task_{Elps2}$ and $P\text{-}TaskM_{Elps2}$ enables tasks and movement tasks to evolve with temporal transitions when they are disabled. Rule $P\text{-}Task_{C}$ deals with the completion of a basic task. If the task is enabled and the timeout is expired, the rule increments the token in the outgoing sequence flow, performs the assignments, disables the task, and resets the timer. Rule $P\text{-}TaskM_{Elps1}$ performs a movement to the next place of the physical layer of the environment when the task is enabled and the next physical

$\mathsf{start}(\mathsf{f},\mathsf{f}') \xrightarrow{\tau} \langle inc(dec(\sigma_f,\mathsf{f}),\mathsf{f}')\rangle$	$\sigma_f(\mathsf{f}) > 0$	(*P-Start*)
$\mathsf{end}(\mathsf{f}) \xrightarrow{\tau} \langle dec(\sigma_f,\mathsf{f})\rangle$	$\sigma_f(\mathsf{f}) > 0$	(*P-End*)
$\mathsf{rcv}(\mathsf{f},\mathsf{m},\mathsf{do.i},\mathsf{f}') \xrightarrow{\tau}$ $\langle inc(dec(\sigma_f,\mathsf{f}),\mathsf{f}'), upd(\mathsf{assign}(\mathsf{v},\mathsf{do.i})), rm(\sigma_{\mathsf{m}},\mathsf{m},\mathsf{v})\rangle$	$\sigma_f(\mathsf{f}) > 0 \wedge$ $\mathsf{v} \in \sigma_m(\mathsf{m})$	(*P-Rcv*)
$\mathsf{snd}(\mathsf{f},\mathsf{m},\mathsf{exp},\mathsf{f}') \xrightarrow{\tau} \langle inc(dec(\sigma_f,\mathsf{f}),\mathsf{f}'), add(\sigma_m,\mathsf{m},\mathsf{v})\rangle$	$\sigma_f(\mathsf{f}) > 0 \wedge$ $eval(\mathsf{exp}) = \mathsf{v}$	(*P-Snd*)
$\mathsf{andSplit}(\mathsf{f},F) \xrightarrow{\tau} \langle inc(dec(\sigma_f,\mathsf{f}),F)\rangle$	$\sigma_f(\mathsf{f}) > 0$	(*P-AndSplit*)
$\mathsf{xorSplit}(\mathsf{f},(\mathsf{exp}',\mathsf{f}')(\mathsf{exp}'',\mathsf{f}'')^*) \xrightarrow{\tau} \langle inc(dec(\sigma_f,\mathsf{f}),\mathsf{f}')\rangle$	$\sigma_f(\mathsf{f}) > 0 \wedge$ $eval(\mathsf{exp}') = true$	(*P-XorSplit*)
$\mathsf{andJoin}(F,\mathsf{f}) \xrightarrow{\tau} \langle inc(dec(\sigma_f,F),\mathsf{f})\rangle$	$\forall \mathsf{f}' \in F \ . \ \sigma_f(\mathsf{f}') > 0$	(*P-AndJoin*)
$\mathsf{xorJoin}(\{\mathsf{f}\} \cup F,\mathsf{f}') \xrightarrow{\tau} \langle inc(dec(\sigma_f,\mathsf{f}),\mathsf{f}')\rangle$	$\sigma_f(\mathsf{f}) > 0$	(*P-XorJoin*)
$\mathsf{eventBased}(\mathsf{f},(\mathsf{m},\mathsf{do.i},\mathsf{f}')(\mathsf{m}',\mathsf{do}'.\mathsf{i}',\mathsf{f}'')^*) \xrightarrow{\tau}$ $\langle inc(dec(\sigma_f,\mathsf{f}),\mathsf{f}'), upd(\mathsf{assign}(\mathsf{v},\mathsf{do.i})), rm(\sigma_m,\mathsf{m},\mathsf{v})\rangle$	$\sigma_f(\mathsf{f}) > 0 \wedge$ $\mathsf{v} \in \sigma_m(\mathsf{m})$	(*P-EventB*)

Fig. 3: Environment-aware BPMN process semantics: events and gateways.

place is reachable. Rule *P-TaskM$_{Elps3}$* enables movement tasks to evolve with temporal transitions when it is enabled, the destination is not reachable, and the current position is not the destination. Rule *P-TaskM$_C$* deals with the completion of the movement task; if the task is enabled and the destination has been reached, the rule increments the outgoing sequence flow, performs the assignments, and disables the task. Rules *P-TaskB* and *P-TaskU* deals with binding and unbinding tasks; if there is a token in the incoming sequence flow of those tasks, the effect of the rules is to increment the number of tokens in the outgoing sequence flow and to decrease the token in the incoming sequence flow, while producing a transition label representing the will of the participant to bind/unbind with the other participant connected by the handshake flow h.

Finally, we comment on the operational rules in Figure 5, dealing with the interleaving of process and collaboration elements. Rule *P-Elps* enables not timed process elements to evolve the process configuration with temporal transitions. Rule *P-Int*$_1$ enables single process elements to evolve the process configuration with silent transitions, instead rule *P-Int*$_2$ enables process elements to evolve the process configuration with temporal transitions $\checkmark$ if no silent actions φ may occur (see maximal progress assumption of timed process calculi [3]). Regarding collaboration elements, rule *C-Prop* propagates the transition from process to collaboration configuration, while *C-Int*$_1$ and *C-Int*$_2$ behave as *P-Int*$_1$ and *P-Int*$_2$, respectively. Notably, rules *P-Int*$_1$ and *C-Int*$_1$ allow single process elements to evolve the configuration, while *P-Int*$_2$ and *C-Int*$_2$ imposes a synchronous triggering of rules producing a temporal transition. Rule *C-Bind* (resp. *C-Unbind*) binds (resp. unbinding) two participants if they are in the same physical

$\mathsf{task}(\mathsf{f},\mathsf{n},\mathsf{exp}_g,A,\mathsf{d},\mathsf{f}') \xrightarrow{\tau} \langle dec(\sigma_f,\mathsf{f}), enab(\sigma_s,\mathsf{n}), set(\sigma_t,\mathsf{n},getD(\mathsf{d}))\rangle$	$\sigma_f(\mathsf{f}) > 0 \wedge \sigma_s(\mathsf{n}) = dis \wedge eval(\mathsf{exp}_g) = true$	$(P\text{-}Task_{Act})$
$\mathsf{task}(\mathsf{f},\mathsf{n},\mathsf{exp}_g,A,\mathsf{d},\mathsf{f}') \xrightarrow{\checkmark} \langle dec(\sigma_t,\mathsf{n})\rangle$	$\sigma_s(\mathsf{n}) = enab \wedge \sigma_t(\mathsf{n}) > 0$	$(P\text{-}Task_{Elps1})$
$\mathsf{task}(\mathsf{f},\mathsf{n},\mathsf{exp}_g,A,\mathsf{d},\mathsf{f}') \xrightarrow{\checkmark} \langle\rangle$	$\sigma_s(\mathsf{n}) = dis$	$(P\text{-}Task_{Elps2})$
$\mathsf{task}(\mathsf{f},\mathsf{n},\mathsf{exp}_g,A,\mathsf{d},\mathsf{f}') \xrightarrow{\tau} \langle inc(\sigma_f,\mathsf{f}'), upd(A), dis(\sigma_s,\mathsf{n}), reset(\sigma_t,\mathsf{n})\rangle$	$\sigma_s(\mathsf{n}) = enab \wedge \sigma_t(\mathsf{n}) = 0$	$(P\text{-}Task_{C})$
$\mathsf{taskM}(\mathsf{f},\mathsf{n},\mathsf{exp}_g,A,\mathsf{exp}_d,\mathsf{f}') \xrightarrow{\tau} \langle dec(\sigma_f,\mathsf{f}), enab(\sigma_s,\mathsf{n})\rangle$	$\sigma_f(\mathsf{f}) > 0 \wedge \sigma_s(\mathsf{n}) = dis \wedge eval(\mathsf{exp}_g) = true$	$(P\text{-}TaskM_{Act})$
$\mathsf{taskM}(\mathsf{f},\mathsf{n},\mathsf{exp}_g,A,\mathsf{exp}_d,\mathsf{f}') \xrightarrow{\checkmark} \langle move(\sigma_p,\sigma_h,\mathsf{p},\mathsf{pp})\rangle$	$\sigma_s(\mathsf{n}) = enab \wedge \mathsf{pp} \in next(PL,\sigma_p,\mathsf{p},eval(\mathsf{exp}_d))$	$(P\text{-}TaskM_{Elps1})$
$\mathsf{taskM}(\mathsf{f},\mathsf{n},\mathsf{exp}_g,A,\mathsf{exp}_d,\mathsf{f}') \xrightarrow{\checkmark} \langle\rangle$	$\sigma_s(\mathsf{n}) = dis$	$(P\text{-}TaskM_{Elps2})$
$\mathsf{taskM}(\mathsf{f},\mathsf{n},\mathsf{exp}_g,A,\mathsf{exp}_d,\mathsf{f}') \xrightarrow{\checkmark} \langle\rangle$	$\sigma_s(\mathsf{n}) = enab \wedge next(PL,\sigma_p,\mathsf{p},eval(\mathsf{exp}_d)) = \emptyset \wedge \neg arrived(\sigma_p,\mathsf{p},eval(\mathsf{exp}_d))$	$(P\text{-}TaskM_{Elps3})$
$\mathsf{taskM}(\mathsf{f},\mathsf{n},\mathsf{exp}_g,A,\mathsf{exp}_d,\mathsf{f}') \xrightarrow{\tau} \langle inc(\sigma_f,\mathsf{f}'), upd(A), dis(\sigma_s,\mathsf{n})\rangle$	$\sigma_s(\mathsf{n}) = enab \wedge arrived(\sigma_p,\mathsf{p},eval(\mathsf{exp}_d))$	$(P\text{-}TaskM_{C})$
$\mathsf{taskB}(\mathsf{f},\mathsf{h},\mathsf{f}) \xrightarrow{\rightarrow\mathsf{h}\leftarrow} \langle inc(dec(\sigma_f,\mathsf{f}),\mathsf{f}')\rangle$	$\sigma_f(\mathsf{f}) > 0$	$(P\text{-}TaskB)$
$\mathsf{taskU}(\mathsf{f},\mathsf{h},\mathsf{f}) \xrightarrow{\leftarrow\mathsf{h}\rightarrow} \langle inc(dec(\sigma_f,\mathsf{f}),\mathsf{f}')\rangle$	$\sigma_f(\mathsf{f}) > 0$	$(P\text{-}TaskU)$

Fig. 4: Environment-aware BPMN process semantics: tasks and movement tasks.

place. Rule *C-Open*$_1$ allows the environmental attributes of the configuration to evolve in an open-world fashion, i.e. independently from the collaboration execution. Rule *C-Open*$_2$, instead, does not allow the environment to evolve independently when there is no elapse of time.

3 Case Study Formalization

To assess our formalization, we propose a case study concerning an emergency response collaboration process, which occurs within a hospital and its surroundings.

$$P \xrightarrow{\checkmark} \langle\rangle \quad notTimedEl(P) = true \quad (P\text{-}Elps)$$

$$\frac{P_1 \xrightarrow{\varphi} \langle\sigma'\rangle}{P_1 \mid P_2 \xrightarrow{\varphi} \langle\sigma'\rangle}\ (P\text{-}Int_1) \qquad \frac{P_1 \xrightarrow{\checkmark} \langle\sigma'\rangle \quad \langle P_2,\sigma'\rangle \xrightarrow{\checkmark} \langle\sigma''\rangle \quad P_1 \mid P_2 \not\xrightarrow{\varphi}}{P_1 \mid P_2 \xrightarrow{\checkmark} \langle\sigma''\rangle}\ (P\text{-}Int_2)$$

$$\frac{\langle P, ENV, \mathsf{p}, \sigma\rangle \xrightarrow{\lambda} \langle\sigma'\rangle}{\langle \mathsf{pool}(\mathsf{p},P), ENV, \sigma\rangle \xrightarrow{\lambda} \langle\sigma'\rangle}\ \lambda \neq \eta \quad (C\text{-}Prop)$$

$$\frac{P_1 \xrightarrow{\rightarrow \mathsf{h} \leftarrow} \langle\sigma'\rangle \quad \langle P_2,\sigma'\rangle \xrightarrow{\rightarrow \mathsf{h} \leftarrow} \langle\sigma''\rangle \quad \sigma_p(\mathsf{p}_1) = \sigma_p(\mathsf{p}_2)}{\mathsf{pool}(\mathsf{p}_1, P_1) \parallel \mathsf{pool}(\mathsf{p}_2, P_2) \xrightarrow{\tau} \langle\sigma'' \cdot bind(\sigma_h, \mathsf{p}_1, \mathsf{p}_2)\rangle}\ (C\text{-}Bind)$$

$$\frac{P_1 \xrightarrow{\leftarrow \mathsf{h} \rightarrow} \langle\sigma'\rangle \quad \langle P_2,\sigma'\rangle \xrightarrow{\leftarrow \mathsf{h} \rightarrow} \langle\sigma''\rangle \quad \sigma_p(\mathsf{p}_1) = \sigma_p(\mathsf{p}_2)}{\mathsf{pool}(\mathsf{p}_1, P_1) \parallel \mathsf{pool}(\mathsf{p}_2, P_2) \xrightarrow{\tau} \langle\sigma'' \cdot unbind(\sigma_h, \mathsf{p}_1, \mathsf{p}_2)\rangle}\ (C\text{-}Unbind)$$

$$\frac{C_1 \xrightarrow{\tau} \langle\sigma'\rangle}{C_1 \parallel C_2 \xrightarrow{\tau} \langle\sigma'\rangle}\ (C\text{-}Int_1) \qquad \frac{C_1 \xrightarrow{\checkmark} \langle\sigma'\rangle \quad \langle C_2,\sigma'\rangle \xrightarrow{\checkmark} \langle\sigma''\rangle \quad C_1 \parallel C_2 \not\xrightarrow{\tau}}{C_1 \parallel C_2 \xrightarrow{\checkmark} \langle\sigma''\rangle}\ (C\text{-}Int_2)$$

$$\frac{\langle C, ENV, \sigma\rangle \xrightarrow{\checkmark} \langle\sigma'\rangle}{[C, ENV, \sigma] \xrightarrow{\checkmark} \langle evolve(\sigma')\rangle}\ (C\text{-}Open_1) \qquad \frac{\langle C, ENV, \sigma\rangle \xrightarrow{\tau} \langle\sigma'\rangle}{[C, ENV, \sigma] \xrightarrow{\tau} \langle\sigma'\rangle}\ (C\text{-}Open_2)$$

Fig. 5: Environment-aware BPMN process and collaboration semantics.

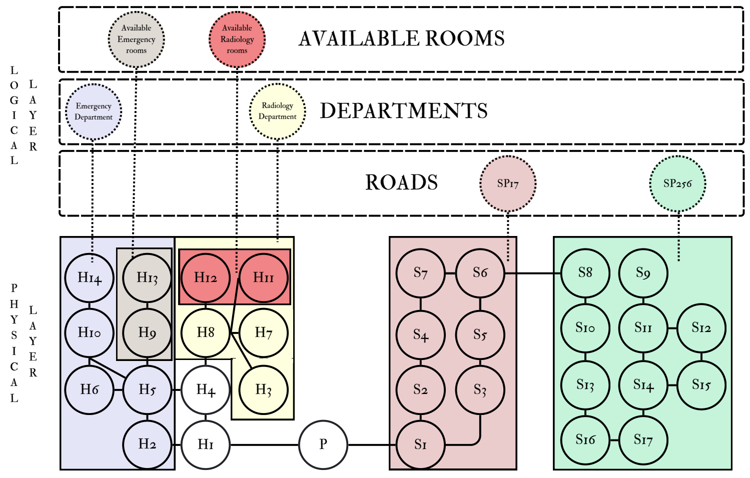

Fig. 6: Case Study Environment Representation

The environment model depicted in Figure 6 overall comprises 32 physical places, 6 logical places and 3 views. The *Departments* view includes the *Emergency Department* and *Radiology Department* logical places. The *Available Rooms* view includes the *Available Emergency Rooms* and *Available Radiology Rooms* logical places. The *Roads* view comprises the *SP17* and *SP256* logical places. The collaboration, see Figure 7, is performed by four participants: *Injured Patient*, *Ambulance*, *Emergency Nurse* and *Emergency Doctor*.

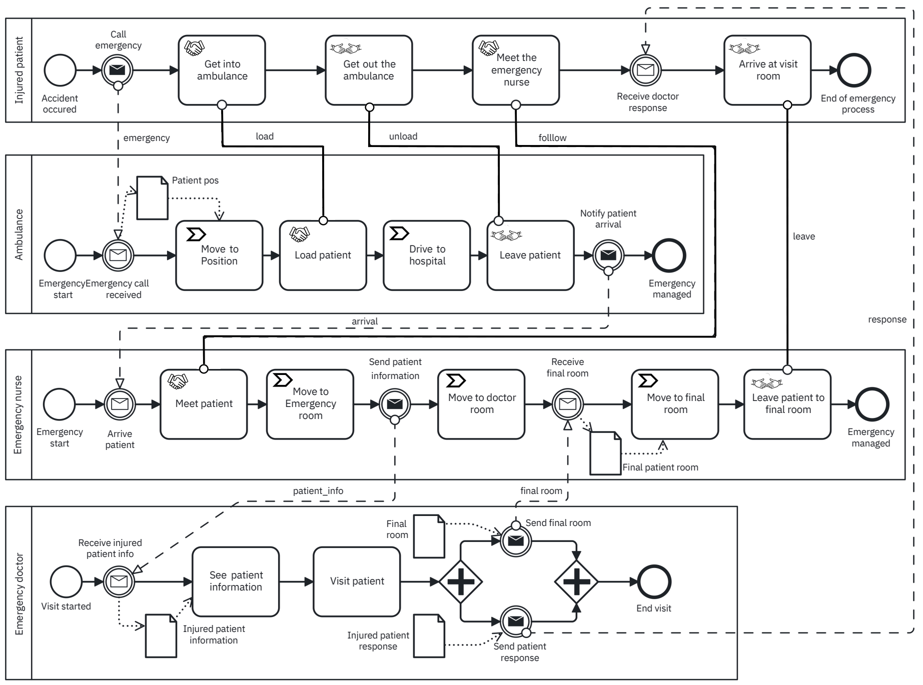

Fig. 7: Emergency response collaboration process

The emergency response collaboration starts when the *Injured Patient* is involved in a car accident and calls an ambulance, providing its position. Upon receiving the call, the *Ambulance* moves to the patient's position. Once arrived, the ambulance picks up the patient with the bind tasks *Get into ambulance* and *Load patient into ambulance* and transports him to the hospital. Once at the hospital, the ambulance leaves the patient to the hospital with an unbind task. The patient is then received by the *Emergency Nurse*, who escorts him to an available emergency room and send his information to the *Emergency Doctor*. The Emergency doctor, after reviewing the patient's information, examines the patient upon his arrival with the nurse. Following the examination, the doctor provides a diagnosis and instructs the nurse on which room the patient should be taken to for further tests. The nurse then escorts the patient to the designated room and leaves him for the necessary analyses.

$$\begin{aligned}
C_{er} &= \mathsf{pool}(\mathsf{p}_{patient}, P_{patient}) \parallel \mathsf{pool}(\mathsf{p}_{ambulance}, P_{ambulance}) \parallel \mathsf{pool}(\mathsf{p}_{nurse}, P_{nurse}) \parallel \mathsf{pool}(\mathsf{p}_{doctor}, P_{doctor})
\end{aligned}$$

$$\begin{aligned}
P_{patient} &= \mathsf{start}(\mathsf{f}'_1, \mathsf{f}_1) \mid \mathsf{snd}(\mathsf{f}_1, \mathsf{m}_{emergency}, \mathsf{v}_{s14}, \mathsf{f}_2) \mid \mathsf{taskB}(\mathsf{f}_2, \mathsf{h}_{load}, \mathsf{f}_3) \mid \mathsf{taskU}(\mathsf{f}_3, \mathsf{h}_{unload}, \mathsf{f}_4) \mid \\
&\quad \mathsf{taskB}(\mathsf{f}_4, \mathsf{h}_{follow}, \mathsf{f}_5) \mid \mathsf{rcv}(\mathsf{f}_5, \mathsf{m}_{response}, \mathsf{do}_{response}.\mathsf{i}_{msg}, \mathsf{f}_6) \mid \mathsf{taskU}(\mathsf{f}_6, \mathsf{h}_{\mathsf{leave}}, \mathsf{f}_7) \mid \mathsf{end}(\mathsf{f}_7)
\end{aligned}$$

$$\begin{aligned}
P_{ambulance} &= \mathsf{start}(\mathsf{f}'_8, \mathsf{f}_8) \mid \mathsf{rcv}(\mathsf{f}_8, \mathsf{m}_{emergency}, \mathsf{do}_{PatientPos}.\mathsf{i}_{position}, \mathsf{f}_9) \mid \mathsf{taskM}(\mathsf{f}_9, \mathsf{n}_{\mathsf{Movetoposition}}, \mathsf{do}_{PatientPos}.\mathsf{i}_{position}, \mathsf{f}_{10}) \mid \\
&\quad \mathsf{taskB}(\mathsf{f}_{10}, \mathsf{h}_{load}, \mathsf{f}_{11}) \mid \mathsf{taskM}(\mathsf{f}_{11}, \mathsf{n}_{Drivetohospital}, \mathsf{pp}_{H1}, \mathsf{f}_{12}) \mid \mathsf{taskU}(\mathsf{f}_{12}, \mathsf{h}_{leave}, \mathsf{f}_{13}) \mid \\
&\quad \mathsf{snd}(\mathsf{f}_{13}, \mathsf{m}_{arrival}, \mathsf{v}_{patientarrival}, \mathsf{f}_{14})\mathsf{end}(\mathsf{f}_{14})
\end{aligned}$$

$$\begin{aligned}
P_{nurse} &= \mathsf{start}(\mathsf{f}'_{15}, \mathsf{f}_{15}) \mid \mathsf{rcv}(\mathsf{f}_{15}, \mathsf{m}_{arrival}, \mathsf{do}_{Info}.\mathsf{i}_{info}, \mathsf{f}_{16}) \mid \mathsf{taskB}(\mathsf{f}_{16}, \mathsf{h}_{follow}, \mathsf{f}_{17}) \mid \\
&\quad \mathsf{taskM}(\mathsf{f}_{17}, \mathsf{n}_{MovetoEmergencyRoom}, (\mathsf{lp}_{availableStudyRooms}.\mathsf{a}_{seats} > 0), \mathsf{A}_1, \mathsf{lp}_{\mathsf{AvailableEmergencyrooms}}, \mathsf{f}_{18}) \mid \\
&\quad \mathsf{snd}(\mathsf{f}_{18}, \mathsf{m}_{patientinfo}, \mathsf{v}_{patientinfo}, \mathsf{f}_{19}) \mid \mathsf{taskM}(\mathsf{f}_{19}, \mathsf{n}_{MovetoDoctorRoom}, \mathsf{pp}_{H4}, \mathsf{f}_{20}) \mid \\
&\quad \mathsf{rcv}(\mathsf{f}_{20}, \mathsf{m}_{finalroom}, \mathsf{do}_{FinalRoom}.\mathsf{i}_{position}, \mathsf{f}_{21}) \mid \mathsf{taskM}(\mathsf{f}_{21}, \mathsf{n}_{MovetoFinalroom}, \mathsf{do}_{FinalRoom}.\mathsf{i}_{position}, \mathsf{f}_{22}) \mid \\
&\quad \mathsf{taskU}(\mathsf{f}_{22}, \mathsf{h}_{leave}, \mathsf{f}_{23}) \mid \mathsf{end}(\mathsf{f}_{23}) \\
&\quad \mathsf{A}_1 = \mathsf{assign}((\mathsf{myplace.a}_{freeSeats} - 1), \mathsf{myplace.a}_{freeSeats})
\end{aligned}$$

$$\begin{aligned}
P_{doctor} &= \mathsf{start}(\mathsf{f}'_{24}, \mathsf{f}_{24}) \mid \mathsf{rcv}(\mathsf{f}_{24}, \mathsf{m}_{patientinfo}, \mathsf{do}_{InjuredPatientInformation}.\mathsf{i}, \mathsf{f}_{25}) \mid \mathsf{task}(\mathsf{f}_{25}, \mathsf{n}_{SeePatientInformation}, \mathsf{d}_1, \mathsf{f}_{26}) \mid \\
&\quad \mathsf{task}(\mathsf{f}_{26}, \mathsf{n}_{VisitPatient}, 1, \mathsf{f}_{27}) \mid \mathsf{andSplit}(\mathsf{f}_{27}, (\mathsf{f}_{28}, \mathsf{f}_{29})) \mid \mathsf{snd}(\mathsf{f}_{28}, \mathsf{m}_{response}, \mathsf{v}_{patientresponse}, \mathsf{f}_{30}) \mid \\
&\quad \mathsf{snd}(\mathsf{f}_{29}, \mathsf{m}_{finalroom}, \mathsf{v}_{\mathsf{pp}_{H8}}, \mathsf{f}_{31}) \mid \mathsf{andSplit}(\mathsf{f}_{30}, (\mathsf{f}_{31}, \mathsf{f}_{32})) \mid \mathsf{end}(\mathsf{f}_{32})
\end{aligned}$$

$$\begin{aligned}
ENV_h &= (PL_h, LL_h) \\
PL_h &= (\{\mathsf{pp}_{H1}, \ldots, \mathsf{pp}_{H14}, \mathsf{pp}_P, \mathsf{pp}_{S1}, \ldots, \mathsf{pp}_{S17}\},\ \{e_1, \ldots, e_{35}\},\ \mathbb{A}_{pp} : \{\mathsf{a}_{zone},\ \mathsf{a}_{purpose},\ \mathsf{a}_{freeSeats}\},\ \mathbb{A}_e : \{\mathsf{a}_{status}\}, \\
&\qquad E = \{(\mathsf{pp}_{H1}, e_1, \mathsf{pp}_{H2}), (\mathsf{pp}_{H2}, e_2, \mathsf{pp}_{H1}), \ldots, (\mathsf{pp}_{S17}, e_{35}, \mathsf{pp}_{S16}), (\mathsf{pp}_{S16}, e_{20}, \mathsf{pp}_{S17}), \} \\
LL_h &= (V_{Departements}, V_{Available\ Rooms}, V_{Roads}, LP_h) \\
LP_h &= [\mathsf{lp}_{EmergencyDepartment} \mapsto \mathsf{a}_{zone} == \mathsf{v}_{hospital}\ and\ \mathsf{a}_{purpose} == \mathsf{v}_{emergency}, \\
&\quad \mathsf{lp}_{Radiology} \mapsto \mathsf{a}_{zone} == \mathsf{v}_{hospital}\ and\ \mathsf{a}_{purpose} == val_{radiology}, \\
&\quad \mathsf{lp}_{AvailableRadiologyRooms} \mapsto \mathsf{a}_{zone} == \mathsf{v}_{hospital}\ and\ \mathsf{a}_{purpose} == \mathsf{v}_{radiology}\ and\ \mathsf{a}_{freeSeats} > 0, \\
&\quad \mathsf{lp}_{AvailableEmergencyRooms} \mapsto \mathsf{a}_{zone} == \mathsf{v}_{hospital}\ and\ \mathsf{a}_{purpose} == \mathsf{v}_{emergency}\ and\ \mathsf{a}_{freeSeats} > 0, \\
&\quad \mathsf{lp}_{SP276} \mapsto \mathsf{a}_{zone} == \mathsf{v}_{street}\ and\ \mathsf{a}_{purpose} == \mathsf{v}_{SP176}, \\
&\quad \mathsf{lp}_{SP17} \mapsto \mathsf{a}_{zone} == \mathsf{v}_{street}\ and\ \mathsf{a}_{purpose} == \mathsf{v}_{SP17}] \\
V_{Departments} &= (\{\mathsf{lp}_{EmergencyDepartment}, \mathsf{lp}_{RadiologyDepartment}\}, \emptyset, \emptyset) \\
V_{AvailableRooms} &= (\{\mathsf{lp}_{AvailableEmergencyRooms}, \mathsf{lp}_{AvailableRadiologyRooms}\}, \{\mathsf{a}_{seats}\}, [\mathsf{a}_{seats} \mapsto (sum\langle \mathsf{a}_{freeSeats}\rangle, occupy\langle \mathsf{a}_{freeSeats}\rangle)]) \\
V_{Roads} &= (\{\mathsf{lp}_{SP17}, \mathsf{lp}_{SP276}\}, \emptyset, \emptyset)
\end{aligned}$$

```
$sum\langle a\rangle(this)$ :                    $occupy\langle a\rangle(this,\ n)$ :
$res = 0;$                                      $forEach$ pp $in$ $\mathscr{P}(this)$
$forEach$ pp $in$ $\mathscr{P}(this)$           $if\ n > pp.a\ \ then\ \ n- = pp.a;\ \ \ pp.a = 0;$
$res+ = pp.a;$                                  $else\ \ pp.a- = n;$
$return\ res;$                                  $return;$
```

Fig. 8: Spatial Textual representation of the university example

In Figure 8, we provide the textual representation of the presented case study. This exemplifies the correspondence between the graphical representation of the environment and the proposed BNF syntax. Note that the edges are undirected in the example, so they are represented by two directed edges with the same edge id, having the same attribute values.

References

1. Corradini, F., Piccioni, J., Re, B., Rossi, L., Tiezzi, F.: On the Interplay Between BPMN Collaborations and the Physical Environment. In: International Conference of Business Process Management, BPM2024. p. 93–110. Springer-Verlag (2024)
2. OMG: Business process model and notation. (BPMN V2.0) (2011)
3. Roever, W.P.d., Hooman, J.: Design and verification in real-time distributed computing: an introduction to compositional methods. In: International Symposium on Protocol Specification, Testing and Verification. p. 37–56. North-Holland Publishing (1989)