Business Processes Meet Spatial Concerns: The sBPMN Verification Framework

Rim Saddem-Yagoubi[1(✉)], Pascal Poizat[2,3], and Sara Houhou[3,4,5]

[1] COSYS-ESTAS, Univ Gustave Eiffel, IFSTTAR, Univ Lille, 59650 Villeneuve d'Ascq, France
rim.saddem@ifsttar.fr
[2] Université Paris Lumières, Université Paris Nanterre, 92000 Nanterre, France
[3] Sorbonne Université, CNRS, LIP6, 75005 Paris, France
{pascal.poizat,sara.houhou}@lip6.fr
[4] Biskra University, LINFI Laboratory, Biskra, Algeria
[5] LIRMM, CNRS & Université de Montpellier, 34095 Cedex 5 Montpellier, France

Abstract. BPMN is the standard for business process modeling. It includes a rich set of constructs for control-flow, inter-process communication, and time-related concerns. However, spatial concerns are left apart while being essential to several application domains. We propose a comprehensive extension of BPMN to deal with this. Our proposal includes an integrated notation, a first-order logic semantics of the extension, and tool-supported verification means through the implementation of the semantics in TLA$^+$. Our tool support and our model database are open source and freely available online.

Keywords: Business processes · Spatial concerns · Formal semantics · Verification · Tool · BPMN · First-order logic · TLA$^+$

1 Introduction

The Business Process Model and Notation (BPMN) [11] is the de facto standard notation for the modeling of business processes. This rich notation includes features to support different aspects to be taken into account when designing a process model. This includes, among others, specifying the control flow, the inter-process communication, and time-related concerns.

Yet, a conventional process model may suit one's needs in a given context and not in other ones (think of rules for allowing, or not, attendance to conferences based on the COVID-related status of the hosting country). A context sensitive approach to business process modeling also offers the ability to adapt the process behaviours to changing contexts [14]. Coined by Schilit and Theimer [16] in 1994, the term "context" has been given a generic definition by Dey [6] in 2001 as *"any information that can be used to characterize the situation of an entity"*. Transferring this definition to the business process management domain, a useful

This work was supported by project PARDI ANR-16-CE25-0006.

definition of a business process context could be *"the minimum set of variables containing all relevant information that impact the design and execution of a business process"* [13]. Saidani and Nurcan further identify in [14] four important kinds of contexts: location, time, resource, and organisation-related contexts. As a contextual factor, location is widely discussed as part of research related to mobile applications and it has wider implications for process management [13].

Without a first-class treatment for the location context in process modelling, one would have to rely on an abstraction of it, *e.g.*, using non-determinism to model conditional branching based on the current state of this context. This is known to yield over-approximation issues. For illustration purposes, let us suppose a process where a robot has to treat rooms in a power-plant or fields in a farm. For this, one relies on a looping workflow pattern, where the robot performs several tasks while there is still at least a place to deal with. Our case study, below, is an instance of this. With support for the location context, it is possible, first, to represent the state of the world (the power-plant rooms or the fields, and their status) and, second, to exit the loop when all places have been treated. Without this support, the fact that there is still, or not, a place to deal with would be modelled using non-determinism. One would then have a possible infinite run, where there is always a place left to treat, making it impossible to perform verification without using bounded analysis (up to some length of runs) or fairness constraints. In line with this, we focus here on the impact of the location-related context in the design and the execution of business processes.

Contributions. Our contributions in this direction are threefold. First, (1.) we propose an extension of the BPMN standard notation in order to take into account location-related (which we call spatial) concerns, (2.) we define a formal execution semantics for this extension, and (3.) we provide the process designers with tools for the automated verification of their extended process models.

With respect to (1.), we support the modeling of control flows based on location constraints, of process mobility, and of the action of processes on their environment. In this work we assume that only the processes at hand have the possibility to change this environment (*i.e.*, the environment cannot independently evolve). We made the choice to extract the specification of the location-related context into a specific structure that is then attached to the process model for verification. This makes the modelling of processes more generic, maintainable, and extensible as one can very simply change the context of use of a process to see the impact on its correctness. This structure, that we call *space structure*, describes all components of a location context: places (called basic locations), possible moves between places, and logical groups of places (called group locations) that can evolve over time and with the action of the processes. All our extensions rely on the standard BPMN extension mechanisms. The usual process design tools can hence be used to model extended processes (in practice, we use Camunda for this). Our extensions can be "forgotten" by verification or execution tools not supporting them. But using our own tooling (see below, 3.) the process designers will be able to go beyond standard BPMN verification and check also models in presence of location extensions.

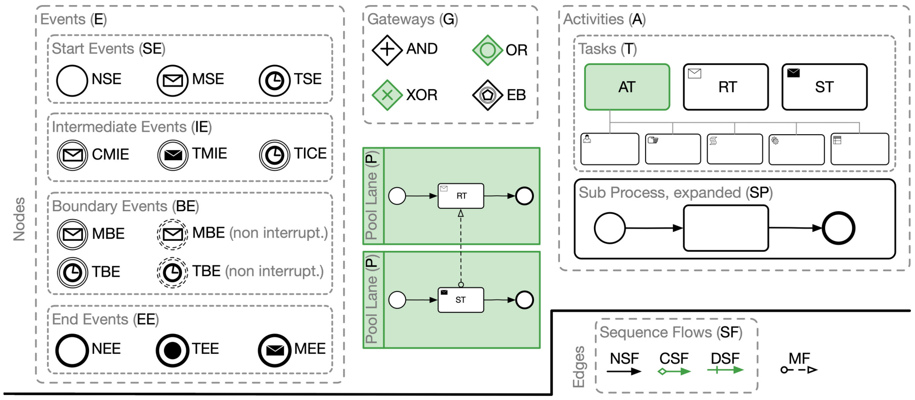


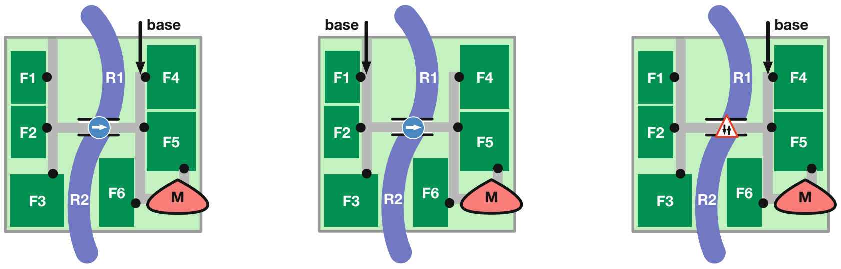
Fig. 1. BPMN subset being supported. Extension points are colored in green. (Color figure online)

With respect to (2.), we follow the approach in [8] and extend its first-order logic execution semantics for BPMN[1] towards taking into account our extensions. This is achieved structurally on the different extension constructs (space structure initial configuration, processes initial locations, extended inclusive and exclusive gateways, tasks for process mobility or for acting on the context).

With respect to (3.), our approach is based on two steps. An extended process model is first transformed into a TLA$^+$ representation of it. Then, given this representation and the TLA$^+$ implementation of the (2.) semantics, the model-checker from the TLA$^+$ tool-suite is used to perform verification of both standard process properties (safety, soundness) and location-related ones. Our tools and the models we use for evaluation are open source and available online [12].

BPMN Primer. The subset of BPMN that we support is given in Fig. 1. The main instrument to control the flow in processes are gateways. Parallel gateways (AND) require that all incoming flows are active and give control to all outgoing flows. Exclusive (XOR) and event-based (EB) gateways only require one of the incoming flows to be active and activate only one of the outgoing flows based either on a condition (for XOR) or some event (for EB, here message reception). The inclusive gateway (OR) has a more complex semantics. First, several outgoing flows can be activated at a time, based on conditions. Second, it requires that the maximum possible number of incoming flows are active (this may require waiting for some time before more incoming flows are activated, even if one is already active). Inter-process message-based communication can be achieved using send tasks (ST) or message throwing events (MEE at the end of a process, TMIE otherwise), and receive tasks (RT) or message catching events (MSE at the start of a process, CMIE otherwise). Activities are subject to boundary events either interrupting or not (the former interrupt the activity, the

[1] As far as time-related elements are concerned, we take the first, non-deterministic, semantics given in [8].



(a) Context 1. (b) Context 2. (c) Context 3.

Fig. 2. Case study – location contexts.

latter activate some flow branch without interrupting the activity). We refer the reader to [11] for more detailed information on the BPMN notation and to [8] for more discussion on the, complex, inclusive gateway.

Case Study. Figures 2 and 3 present the case study we use to illustrate our proposal and to perform verification. The outcomes of verification on more examples are synthesized in Sect. 3. This model is a simplified version of a collaborative process model where a controller (Controller), a crop planting robot (Planter), and a watering robot (Sprayer, not taken into account here) have to plant crops and water several fields spread over both banks of a river.

The context of the collaboration is the river and its banks, with three different instances given in Figs. 2a–2c. Places (basic locations) are either fields (F1 to F6), river (R1 and R2), bridge (B), or mountain (M). There is also the base (base) from which our collaboration peers will operate and move. Over these places we may define logical groups of locations (of simply, group locations) for the places where to plant crops (toPlant), the places to water (toWater), the places with crops (planted), and the watered places (watered). Finally, we have possible moves around the location context, based on roads and bridge. All together, basic locations, group locations, and possible moves, make up possible space structures to be associated to location-aware processes. Please note that in practice the location structures are of course not denoted using pictures but rather as a set of information extending the BPMN element for the process collaboration:

```
<bpmn:collaboration id="s006Robots2"><bpmn:extensionElements>
      <camunda:properties>
        <camunda:property name="base-locations"
                          value="[base,F1,F2,F3,F4,F5,F6,R1,R2,B,M]" />
        <camunda:property name="group-locations"
                          value="[toPlant,planted,toWater,watered]" />
        <camunda:property name="transitions"
                          value="[(base,F1),(F1,base),(F1,F2),(F2,F1),(F2,F3),...,(M,F6)]" />
        <camunda:property name="initial-space"
                          value="{toPlant:[F1,F2,F3,F4,F5,F6],
                                  planted:[], toWater:[], watered:[]}" />
      </camunda:properties>
</bpmn:extensionElements></bpmn:collaboration>
```

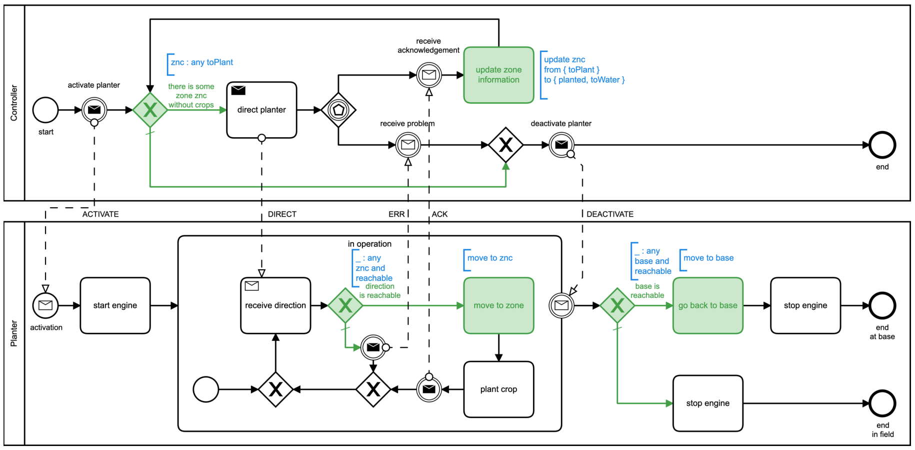


Fig. 3. Crop planting collaboration.

In Fig. 3 we have a BPMN collaboration that has been extended with the extensions we propose. In green we show which elements are extended (in addition to the whole collaboration, as seen before, and the processes' pool lanes extended to denote where the processes start). In blue we use notes to give, as comments, extensions that are indeed also stored as extension elements in the XML source BPMN file. We remind that since we use BPMN extension mechanisms, these extensions can be directly entered in BPMN design tools. The whole model (diagram and source file, with extensions) is available from [12].

The controller starts by activating the planter. It then checks if there are still some places where to plant crops. If so it sets znc to be any of these. In our extension checking some location-related constraint and setting accordingly a location variable is done at once using a *space condition*, znc: any toPlant (set znc to be any of the places in group toPlant). Then the controller sends a message to the planter ordering it to get there and waits to receive either an acknowledgement or the information that there is some problem. In the first case the controller registers that the place is now planted using an *update action*, removing znc from group toPlant and adding it to groups planted and toWater. If there is a problem, then the controller interrupts the planter and stops.

On its side, the planter starts upon activation by the controller and, after starting its engine, it enters the in operation sub-process. In this sub-process, it loops on receiving an order to move (it has then access to the shared znc information), checking if znc can be reached from its current location (using reachable, denoting places that are reachable, in the space condition _: any znc and reachable), and accordingly either moving to znc using a *move* action and sending an acknowledgement to the controller, or signalling a problem. Upon interruption by the controller, the loop stops and the planter tries to get back to the base (again, using a space condition to check if this is possible).

We were able to model spatial concerns in our collaboration. But what about correctness? Can the collaboration processes reach termination (black circles in the rightmost part of Fig. 3)? Are all fields planted at the end? Let us consider the contexts. If the planter starts on the right bank then the left one cannot be planted due to the one-way bridge (Fig. 2a). If the planter starts on the left bank (Fig. 2b) or if the bridge is two-way (Fig. 2c) then all fields can be planted.

This case study illustrates several points. First, a process model can operate on (or be adapted to) different space structures. Conversely, a space structure can be used with many processes (we have used one with several variants of the crop planting collaboration). This is made possible by separating the specifications of the context and of the processes, and then integrating them. Further, we have seen that, switch two variants of a process or two variants of a space structure and correctness may no longer hold. In the sequel we will see how to give a formal semantics to our extensions and support their verification using tools.

Outline. The formal part of the paper is developed in Sect. 2, addressing the presentation of the models underlying the semantics, and then the semantics itself. The implementation of the semantics in TLA$^+$, verification, and evaluation are then presented in Sect. 3. This section also includes a short introduction to the TLA$^+$ language and verification framework. Related work is given in Sect. 4, and we end with conclusions and perspectives in Sect. 5.

2 Formal Semantics

In this section, we first give the formal models for extended BPMN diagrams. Then, we present their formal semantics. It follows the "token game" given (in natural language) in Chap. 13 of the standard [11] and builds on the semantics for basic BPMN given in [8]. Hence, we will focus on points of extensions only.

2.1 Formal Models for Space BPMN

As we have seen earlier, space structures represent contexts with base locations, group locations, and possible moves. Hence the following definition.

Definition 1 (Space Structure). *A space structure is a tuple $\mathbb{S} = (\mathbb{B}, \mathbb{G}, \rightarrow)$ where $\mathbb{B}$ is the set of base locations, $\mathbb{G}$ is the set of group locations, and $\rightarrow \subseteq \mathbb{B} \times \mathbb{B}$ denotes all possible moves between base locations. Further, we require that $\mathbb{B} \cap \mathbb{G} = \emptyset$ and $(b, b) \in \rightarrow$ for every b in $\mathbb{B}$.*

Space structures may represent different kinds of location-based contexts such as countries and administrative or epidemiological statuses, offices and security levels, or agricultural environments as in our case study where, $\mathbb{B} = \{\mathsf{F1}, \ldots, \mathsf{M}\}$, $\mathbb{G} = \{\mathsf{toPlant}, \mathsf{toWater}, \mathsf{planted}, \mathsf{watered}\}$, and $\{(\mathsf{F2}, \mathsf{F2}), (\mathsf{F2}, \mathsf{F5})\} \subset \rightarrow$. The set of all locations is denoted by $\mathbb{L} = \mathbb{B} \cup \mathbb{G}$, $\rightarrow^*$ denotes the transitive closure of $\rightarrow$, and $b_1 \rightarrow b_2$ denotes that $(b_1, b_2) \in \rightarrow$.

Notation. In the sequel, definitions are taken given a space structure $\mathbb{S} = (\mathbb{B}, \mathbb{G}, \rightarrow)$ and a set of variables V such that $\mathrm{V} \cap \mathbb{L} = \emptyset$.

Space formulas are used to denote a subset of the base locations (see their interpretation in Def. 10). They are central to our extension as they are used both in conditions on conditional sequence flows (CSF) and in mobility actions.

Definition 2 (Space Formula). *The set of (space) formulas over* $\mathbb{S}$ *and* V*, denoted by* $Form$*, is the smallest set generated by* $\{\mathsf{true}, v, b, g, \mathsf{not}\ f, f_1\ \mathsf{or}\ f_2, \mathsf{here}, \mathsf{reachable}\}$ *for* $v \in \mathrm{V}$*,* $b \in \mathbb{B}$*,* $g \in \mathbb{G}$*, and* $f, f_1, f_2 \in Form$*. Further,* $f_1\ \mathsf{and}\ f_2$ *is defined as* $\mathsf{not}\ (\mathsf{not}\ f_1\ \mathsf{or}\ \mathsf{not}\ f_2)$*.*

Space actions are extensions of BPMN (abstract) tasks that can represent either a move or the update of the context by some process. Additionally, we use a specific pass action for tasks that are not related to the spatial concerns.

Definition 3 (Space Action). *The set of (space) actions over* $\mathbb{S}$ *and* V*, denoted by* $Action$*, is the smallest set generated by* $\{\mathsf{move}(f), \mathsf{update}(u, G^-, G^+), \mathsf{pass}\}$ *for* $f \in Form$*,* $u \in \mathrm{V}$*,* $G^- \subseteq \mathbb{G}$*,* $G^+ \subseteq \mathbb{G}$*, and* $G^- \cap G^+ = \emptyset$*.*

Informally (the formal semantics is given in Def. 11), $\mathsf{move}(f)$ denotes that the process of interest moves to a location satisfying f and $\mathsf{update}(u, G^-, G^+)$ denotes the update of the context by removing some locations (retrieved by evaluating u) from all groups in G^- and adding them to all groups in G^+.

We may now give definitions for processes, which are represented using graphs with typed nodes and edges. Types correspond to the BPMN elements (see Fig. 1, *e.g.*, XOR for exclusive gateways): $T_{Nodes} = \{AT, RT, ST, NSE, MSE, TSE, CMIE, TMIE, TICE, MBE, TBE, NEE, TEE, MEE, AND, OR, XOR, EB, SP, P\}$ and $T_{Edges} = \{NSF, CSF, DSF, MF\}$. Definitions 4 and 5 are taken (and restructured) from [8]. Definition 6 integrates our extensions.

Definition 4 (Graph). *A graph is a tuple* $\rho = (N, E, source, target)$ *where* N *is the set of nodes,* E *(with* $N \cap E = \emptyset$*) is the set of edges, and* $source/target : E \rightarrow N$ *denote the source/target of an edge.*

Definition 5 (BPMN Graph). *A BPMN graph is a tuple* $\widehat{\rho} = (\rho, cat_N, cat_E, R, \mathbb{M}, msg_t)$ *where* ρ *is a graph,* $cat_N : N \rightarrow T_{Nodes}$ *gives the type of a node (*N^T *denoting the set* $\{n \in N \mid cat_N(n) \in T\}$*),* $cat_E : E \rightarrow T_{Edges}$ *gives the type of an edge (*E^T *denoting the set* $\{e \in E \mid cat_E(e) \in T\}$*),* $R : N^{\{SP,P\}} \rightarrow 2^{N \cup E}$ *gives the set of nodes and edges which are directly contained in a container (process or sub-process),* $\mathbb{M}$ *is the set of message types, and* $msg_t : E^{\{MF\}} \rightarrow \mathbb{M}$ *gives the message associated to a message flow.*

Definition 6 (Space BPMN Graph). *A space BPMN graph is a tuple* $\rho^\circ = (\widehat{\rho}, \mathbb{S}, \mathrm{V}, cvar, ckind, ccond, \prec, act)$ *where* $\widehat{\rho}$ *is a BPMN graph,* $\mathbb{S}$ *is a space structure,* V *is a set of variables,* $cvar : E^{\{CSF\}} \rightarrow \mathrm{V}$*,* $ckind : E^{\{CSF\}} \rightarrow \{\mathsf{any}, \mathsf{all}\}$*, and* $ccond : E^{\{CSF\}} \rightarrow Form$ *are total functions that assign respectively a* *variable, a kind (denoting how to associate the interpretation of a space formula to a condition variable), and a space formula to each conditional edge, $\prec \subseteq E^{\{CSF\}} \times E^{\{CSF\}}$, is a relation ordering the conditional sequence flows outgoing from a conditional gateway, and $act : N^{\{AT\}} \rightarrow Action$ is a total function that assigns an action to each abstract task.*

2.2 Semantics for Space BPMN: States and Initial State

The semantics relies on the notion of *state* of the space BPMN graph, which is defined in terms of state markings, edge markings, and space configuration.

Notation. In the sequel, definitions are taken given a space BPMN graph $\rho^\circ = (\widehat{\rho}, \mathbb{S}, \mathrm{V}, cvar, ckind, ccond, \prec, act)$ with $\mathbb{S} = (\mathbb{B}, \mathbb{G}, \rightarrow)$ and $\mathbb{L} = \mathbb{B} \cup \mathbb{G}$.

Definition 7 (Space Configuration). *A (space) configuration c is a couple of substitutions (σ, subs), σ being a variable to base locations substitution ($\sigma : \mathrm{V} \rightarrow 2^{\mathbb{B}}$) and* subs *being a group to base locations substitution (*$\mathrm{subs} : \mathbb{G} \rightarrow 2^{\mathbb{B}}$*).*

The set of all configurations (of a space BPMN graph ρ°) is denoted by $Configs$. We note $\sigma^\perp$ the empty variable configuration ($\forall v \in \mathrm{V},\ \sigma^\perp(v) = \emptyset$). Given a configuration $c = (\sigma, \mathrm{subs})$, $\sigma[v \mapsto B]$, denotes the update of σ by associating B to v (*i.e.*, $\sigma[v \mapsto B]$ is $\sigma(v')$ for each v' in $\mathrm{V} \setminus \{v\}$ and B for v). Accordingly, $\mathrm{subs}[g \mapsto B]$, denotes the update of subs by associating B to g.

Among the variables, we have process locations, *i.e.*, we have a variable $loc_p \in \mathrm{V}$ for each p in $N^{\{P\}}$, and $\sigma(loc_p)$ gives the location of a process p. A configuration encompasses (in σ) the current values of variables – including the location of processes – and (in subs) the state of the relation between group and base locations. To become a state, a configuration is completed with the current marking of the BPMN graph nodes and edges.

Note that here we assume a model of shared variables (*e.g.* the znc variable in our example). This is driven by formalization convenience. Local variables with values exchanged using communication (messages) is part of our perspectives.

Definition 8 (State). *A state is a triple $s = (mn, me, c)$ where $mn : N \rightarrow \mathbb{N}$ (node marking), $me : E \rightarrow \mathbb{N}$ (edge marking), and c is a configuration.*

The set of all states (of a space BPMN graph ρ°) is denoted by $States$. One of them is the state in which the graph starts its execution: its initial state.

Definition 9 (Initial state). *The initial state of a space BPMN Graph ρ° is a state $s_o = (mn_0, me_0, c_0 = (\sigma_0, \mathrm{subs}_0))$ such that only processes and their start events are marked ($\forall n \in N, mn_0(n) = 1$ if $\exists p \in N^{\{P\}}, n \in N^{\{NSE,MSE,TSE\}} \cap R(p)$, and 0 otherwise), no edge is marked ($\forall e \in E, me_0(e) = 0$), and σ_0 is set only for the start locations of processes ($\forall v \in \mathrm{V}$, either $\exists p \in N^{\{P\}}, \exists loc_{p,0} \in \mathbb{B}, v = loc_p \wedge \sigma_0(v) = \{loc_{p,0}\}$ or $\sigma_0(v) = \emptyset$).*

The initial locations for processes (the $loc_{p,0}$) and subs_0 are *parameters of the model* and, hence, of its verification. In our example, for the context in Fig. 2b, we had all fields to plant and the planter starting in F1 (or in base, with base $\rightarrow$ F1). This means that the configuration in the initial state is such that $\sigma_0(loc_{\mathsf{Planter}}) = \{\mathsf{F1}\}$ (or $\{\mathsf{base}\}$) and $\mathrm{subs}_0(\mathsf{toPlant}) = \{\mathsf{F1}, \ldots, \mathsf{F6}\}$.

2.3 Semantics for Space BPMN: Formulas and Actions

Having defined configurations, it is now possible to give a semantics to formulas and actions. Let us begin with the former.

Definition 10 (Formula Interpretation). *The interpretation of a formula f with reference to a configuration $c = (\sigma, \mathrm{subs})$ and a process p, is denoted by $||f||_c^p$ ($||\ \_\ ||_{\_}^{\_} : Form \times N^{\{P\}} \times Configs \to 2^{\mathbb{B}}$) and defined as: $||\mathsf{true}||_c^p = \mathbb{B}$ (all base locations), $||v||_c^p = \sigma(v)$ (value of a variable), $||b||_c^p = \{b\}$ (base location), $||g||_c^p = \mathrm{subs}(g)$ (current locations of a group), $||\mathsf{not}\ f||_c^p = \mathbb{B} \setminus ||f||_c^p$ (difference), $||f_1\ \mathsf{or}\ f_2||_c^p = ||f_1||_c^p \cup ||f_2||_c^p$ (union), $||\mathsf{here}||_c^p = \sigma(loc_p)$, and $||\mathsf{reachable}||_c^p = \{b \in \mathbb{B} \mid \sigma(loc_p) \to^* b\}$ (reachable locations).*

In our example we use formula znc and reachable in the planter. Its interpretation is the intersection of $\sigma\{\mathsf{znc}\}$ and of the locations that are reachable from the planter location. This is not empty only if znc is reachable, which is what we want to express. Let us now come to the actions.

Definition 11 (Actions Evaluation). *The evaluation of an action a over a configuration $c = (\sigma, \mathrm{subs})$ and a process p is denoted by $[[a]]_c^p$ ($[[\ \_\ ]]_{\_}^{\_} : Action \times N^{\{P\}} \times Configs \to Configs$) and defined as follows:*

- $[[\mathsf{move}(f)]]_c^p = c'$ *where c' is c if $||f\ \mathsf{and}\ \mathsf{reachable}||_c^p = \emptyset$ and c' is $(\sigma[loc_p \mapsto \{b\}], \mathrm{subs})$ with $\{b \in ||f\ \mathsf{and}\ \mathsf{reachable}||_c^p\}$ otherwise.*
- $[[\mathsf{update}(u, G^-, G^+)]]_c^p = (\sigma, \mathrm{subs}')$ *where $\mathrm{subs}'(g)$ is $\mathrm{subs}(g) \setminus \sigma(u)$ if $g \in G^-$, $\mathrm{subs}(g) \cup \sigma(u)$ if $g \in G^+$, and $\mathrm{subs}(g)$ otherwise (remind that $G^- \cap G^+ = \emptyset$)*
- $[[\mathsf{pass}]]_c^p = c$

Action move(f) does nothing if f corresponds to no reachable locations satisfying it, otherwise one of the locations is chosen and the process location is updated accordingly. Action update(u, G^-, G^+) updates groups in G^- and G^+ respectively adding/removing locations in u. Action pass does nothing.

In our example the planter tries to move to its base at the end. Using condition _: base and reachable, it checks before that this is indeed possible. Without checking this (suppose there is no exclusive gateway) trying move(base) if the base is not reachable would make the process block on this task.

2.4 Semantics for Space BPMN: Execution Semantics

In order to maintain traceability with the standard [11], we use a token-based approach. The movement of tokens is based on node types. We define an execution model based on two predicates (St, Ct) for each node type which correspond to, respectively, the enabling of the node to start its execution, and the enabling of the node to complete its execution. When not specified, the predicate is *true*.

The formal execution semantics is given in Table 1. The semantics of the elements in Fig. 1 and not in Table 1 is kept from [8]. We consider that mn and mn' (resp. me and me') denote two successive markings of a node (resp. edge)

Table 1. Space BPMN semantics (extensions to [8]).

$n \in N^{\{XOR\}}$	$Ct(n) \overset{def}{\equiv} \exists ei \in in^{SF}(n), me(ei) \geq 1 \wedge me'(ei) = me(ei) - 1$ $\wedge\ \exists eo_1 \in out^{\{CSF\}}(n), \|\|ccond(eo_1)\|\|_c^{procOf(n)} \neq \emptyset$ $\wedge\ \forall eo_2 \in out^{\{CSF\}}(n), eo_2 \prec eo_1 \Rightarrow \|\|ccond(eo_2)\|\|_c^{procOf(n)} = \emptyset$ $\wedge\ me'(eo_1) = me(eo_1) + 1$ $\wedge\ \exists v \in \mathrm{V}, v = cvar(eo_1)$ $\wedge\ \mathrm{ckind}(v) = \mathsf{all}$ $\Rightarrow \sigma' = \sigma[v \mapsto \|\|ccond(eo_1)\|\|_c^{procOf(n)}]$ $\wedge\ \mathrm{ckind}(v) = \mathsf{any}$ $\Rightarrow \exists x \in \|\|ccond(eo_1)\|\|_c^{procOf(n)}, \sigma' = \sigma[v \mapsto x]$ $\wedge\ \triangle(\{ei, eo_1\}) \wedge \Xi_{\mathrm{subs}}$ $\vee \wedge \forall eo_1 \in out^{\{CSF\}}(n), \|\|ccond(eo_1)\|\|_c^{procOf(n)} = \emptyset$ $\wedge\ \exists eo_2 \in out^{\{DSF\}}(n), me'(eo_2) = me(eo_2) + 1$ $\wedge\ \triangle(\{ei, eo_2\}) \wedge \Xi^{\sigma}_{\mathrm{subs}}$
$n \in N^{\{OR\}}$	$Ct(n) \overset{def}{\equiv} In^{+}(n) \neq \emptyset$ $\wedge\ \forall ei \in In^{+}(n), me'(ei) = me(ei) - 1$ $\wedge\ \forall ez \in In^{-}(n),$ $\forall ee \in (Pre_E(n, ez) \setminus ignore_E(n)), me(ee) = 0$ $\wedge\ \forall nn \in (Pre_N(n, ez) \setminus ignore_N(n)), mn(nn) = 0$ $\wedge\ \exists es \subseteq out^{\{CSF\}}(n) \wedge es \neq \emptyset$ $\wedge\ \nexists eo \in out^{\{CSF\}}(n) \setminus es, \|\|ccond(eo)\|\|_c^{procOf(n)} \neq \emptyset$ $\wedge\ \forall eo \in es, \|\|ccond(eo)\|\|_c^{procOf(n)} \neq \emptyset$ $\wedge\ me'(eo) = me(eo) + 1$ $\wedge\ \exists v \in \mathrm{V}, v = cvar(eo)$ $\wedge\ \mathrm{ckind(v)} = \mathsf{all}$ $\Rightarrow \sigma' = \sigma[v \mapsto \|\|ccond(eo)\|\|_c^{procOf(n)}]$ $\wedge\ \mathrm{ckind(v)} = \mathsf{any}$ $\Rightarrow \exists x \in \|\|ccond(eo)\|\|_c^{procOf(n)}, \sigma' = \sigma[v \mapsto x]$ $\wedge\ \triangle(In^{+}(n) \cup es) \wedge \Xi_{\mathrm{subs}}$ $\vee \wedge \forall eo_1 \in out^{\{CSF\}}(n), \|\|ccond(eo_1)\|\|_c^{procOf(n)} = \emptyset$ $\wedge\ \exists eo_2 \in out^{\{DSF\}}(n), me'(eo_2) = me(eo_2) + 1$ $\wedge\ \triangle(In^{+}(n) \cup \{eo_2\}) \wedge \Xi^{\sigma}_{\mathrm{subs}}$
$n \in N^{\{AT\}}$	$St(n) \overset{def}{\equiv} \exists ei \in in^{SF}(n), me(ei) \geq 1$ $\wedge\ me'(ei) = me(ei) - 1 \wedge mn'(n) = mn(n) + 1 \wedge \triangle(\{n, ei\}) \wedge \Xi^{\sigma}_{\mathrm{subs}}$
	$Ct(n) \overset{def}{\equiv} mn(n) \geq 1$ $\wedge\ mn'(n) = mn(n) - 1$ $\wedge\ \forall eo \in out^{SF}(n), me'(eo) = me(eo) + 1$ $\wedge\ c' = [[act(n)]]_c^{procOf(n)}$ $\wedge\ act(n) = \mathsf{move}(f) \Rightarrow \|\|f \text{ and reachable}\|\|_c^{procOf(n)} \neq \emptyset$ $\wedge\ act(n) = \mathsf{pass} \wedge \triangle(\{n\} \cup out^{SF}(n)) \wedge \Xi^{\sigma}_{\mathrm{subs}}$ $\vee\ act(n) = \mathsf{move}(f) \wedge \triangle(\{n\} \cup out^{SF}(n)) \wedge \Xi_{\mathrm{subs}}$ $\vee\ act(n) = \mathsf{update}(u, G^{-}, G^{+}) \wedge \triangle(\{n\} \cup out^{SF}(n)) \wedge \Xi^{\sigma}$

in the execution semantics. Accordingly $c = (\sigma, \mathrm{subs})$ and $c' = (\sigma', \mathrm{subs}')$ denote two successive configurations. $\triangle$ is a predicate that denotes marking equality but for nodes and edges given as parameter, $\triangle(X)$ means "*nothing changes but for*

X": $\triangle(X) \equiv \forall n \in N \setminus X, mn'(n) = mn(n) \wedge \forall e \in E \setminus X, me'(e) = me(e)$. In the same way, we define $\Xi^\sigma \equiv \sigma' = \sigma$, $\Xi_{\text{subs}} \equiv \text{subs}' = \text{subs}$, and $\Xi^\sigma_{\text{subs}} \equiv \Xi^\sigma \wedge \Xi_{\text{subs}}$.

In the semantics we use helper functions: $in/out : N \rightarrow 2^E$ give the incoming/outgoing edges of a node, $in(n) = \{e \in E \mid target(e) = n\}$ and $out(n) = \{e \in E \mid source(e) = n\}$. A family of functions in^T (resp. out^T) : $N \rightarrow 2^E$ is used to combine in (resp. out) with E^T, $in^T(n) = in(n) \cap E^T$ and $out^T(n) = out(n) \cap E^T$. The type set for sequence flows, SF, is defined as $\{NSF, CSF, DSF\}$. $procOf : N \rightarrow N^{\{P\}}$ gives the container process of a given node, $procOf(n) = p$ if and only if $n \in R^+(p)$, with R^+ being the transitive closure of R. Further, to deal with the complex semantics of the OR gateway, and formalize in a correct way its definition in [11], we use some more functions: $In^- : N \rightarrow E$ gives the unmarked incoming edges of a node: $In^-(n) = \{e \in in^{SF}(n) \mid me(e) = 0\}$. $In^+ : N \rightarrow E$ gives the marked incoming edges of a node: $In^+(n) \overset{def}{=} \{e \in in^{SF}(n) \mid me(e) \geq 1\}$. $Pre_N : N \times E \rightarrow 2^N$, gives the predecessor nodes of an edge such that n^{pre} is in $Pre_N(n, e)$ if there is a path from n^{pre} to e that never visits n. Accordingly, $Pre_E : N \times E \rightarrow 2^E$ gives predecessor edges. $ignore_E : N \rightarrow 2^E$ gives the predecessor edges of the marked incoming edges of a given node: $ignore_E(n) = \bigcup_{e^+ \in In^+(n)} Pre_E(n, e^+)$. $ignore_N : N \rightarrow 2^N$ gives the predecessor nodes of the marked incoming edges of a given node: $ignore_N(n) = \bigcup_{e^+ \in In^+(n)} Pre_N(n, e^+)$.

Let us now describe Table 1.

Space Formulas and Gateways. Gateways are atomic hence define only the Ct predicate. A XOR gateway may complete by choosing one of its outgoing conditional flows eo_1 whose formula is satisfied (its interpretation, Def. 10, is not empty). In case there are several such transitions, we follow the BPMN standard and use ordering ($\prec$) so that only the first one is activated. If there is no such flow, the default sequence flow outgoing from the gateway (if it exists) is activated. When a conditional flow is activated, its variable is updated at the same time. Depending on the condition kind (any or all), either a single location from the formula interpretation or all of them are taken to update $\sigma(v)$.

An OR gateway has some differences *wrt.* a XOR gateway. First it requires that no more ongoing flow ez can be activated before completing (lines 3–5 of Ct). Second, all outgoing conditional flows whose conditions are satisfied are activated (lines 6–8 of Ct are used to get the maximal set es of such flows).

Space Actions and Abstract Tasks. Actions are associated to abstract tasks which can start if at least one of their incoming flows is active. The completion of these tasks depends on the evaluation of their action (Def. 11) in order to change the configuration (c to c'), either σ for moves or subs for updates. For move(f), we require that there is at least one reachable location in the interpretation of f.

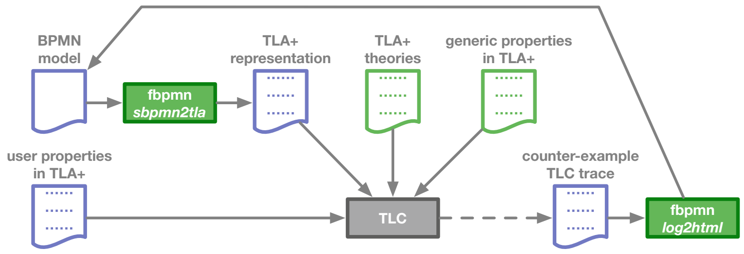


Fig. 4. Tool support (blue: model-specific, green: non model-specific).

3 Implementation and Verification

In this section, we present the implementation of our approach in a tool suite, sBPMN, enabling the verification of extended BPMN models and the animation of counter-examples. sBPMN is a part in a more general project about BPMN verification [12]. It works as presented in Fig. 4.

The user first designs an extended BPMN model using any modeler with support for BPMN extension mechanisms (we use Camunda). This model corresponds to Definition 5 with extensions information for Definition 6 (extensions to BPMN) and Definition 9 (initial state parameter). In practice, the space structure and the initial value of the subs relation are associated to the process diagram (as shown in page 5), initial locations are associated to processes, and conditional information and actions are associated respectively to conditional flows and to abstract tasks. A TLA$^+$ representation of this model is retrieved using the fbpmn command with option sbpmn2tla. Given this representation, the TLA$^+$ implementation of the semantics, and the TLA$^+$ encoding of generic and user given model-specific properties, the TLC model checker from the TLA$^+$ tool box (freely available at http://lamport.azurewebsites.net/tla/tla.html) is used to perform verification. If some properties are not satisfied, TLC outputs a counter-example trace that is then transformed using the fbpmn command with option log2html into an HTML+JS page enabling the user to animate the counter example on the BPMN model, step by step, to understand the error(s). We provide users with a script, sfbpmn-check, that performs all steps in Fig. 4 in a transparent way (see [12], under scripts/).

3.1 Implementing the Semantics and Encoding Models in TLA$^+$

TLA$^+$ [9] is a formal specification language based on untyped Zermelo-Fraenkel set theory for specifying data structures, and on the temporal logic of actions (TLA) for specifying dynamic behaviors. TLA$^+$ allows one to specify symbolic transition systems with variables and *actions*. An action is a transition predicate between a state and a successor state. It is an arbitrary first-order predicate with quantifiers, set and arithmetic operators, and functions, and where quoting denotes the value of variables in the successor state.

This is in phase with our St and Ct predicates. Further, the expression and action fragment of TLA$^+$ contains first-order logic, that we use to express the semantics. Hence, the encoding of the semantics in TLA$^+$ is straightforward (up to the TLA$^+$ concrete syntax of course). For example, the TLA$^+$ implementation for the semantics of XOR nodes is given in Fig. 5. The comprehensive TLA$^+$ theories for the semantics is available at [12] under `theories/stla`.

```
MODULE XOR
$xor\_complete(n) \triangleq$
  $\land\ CatN[n] = ExclusiveOr$
  $\land\ \exists\, ei \in intype(SeqFlowType, n):$
    $\land\ edgemarks[ei] \geq 1$
    $\land\ edgemarks' = [edgemarks\ \text{EXCEPT}\ ![ei] = @ - 1]$
  $\land\ \lor\ \exists\, eo1 \in outtype(\{ConditionalSeqFlow\}, n):$
        ...
    $\lor\ \land\ \forall\, eo1 \in outtype(\{ConditionalSeqFlow\}, n):$
            $evalF(sigma, subs, ProcessOf(n), cCond[eo1]) = \{\}$
      $\land\ \exists\, eo2 \in outtype(\{DefaultSeqFlow\}, n):$
            $edgemarks' = [edgemarks\ \text{EXCEPT}\ ![eo2] = @ + 1]$
      $\land\ \text{UNCHANGED}\ nodemarks \land \text{UNCHANGED}\ subs \land \text{UNCHANGED}\ sigma$
```

Fig. 5. Semantics of XOR nodes in TLA$^+$ (part of).

3.2 Properties and Verification

TLC is an explicit-state model checker that checks both safety and liveness properties specified in LTL. This logic includes operators $\Box$ and $\Diamond$ that respectively denote that, in all executions, a property F must always hold ($\Box F$) or that it must hold at some instant in the future ($\Diamond F$). In our framework, properties are either properties that are generic to all process models, or user given properties specific to a given process model (see Fig. 6).

For the former, we reuse the TLA$^+$ implementation given in [8] for the properties defined by [5]: (1) safety, (2) soundness and (3) message-relaxed soundness. A collaboration is safe if no sequence flow holds more than one token. A collaboration is sound if all processes are sound and there are no undelivered messages, a process being sound if it can reach a state where nothing but its events are active. Finally, message-relaxed soundness is soundness without taking into account undelivered messages (tokens on message flows). For the model-specific properties, the user can rely on the elements in states (Def. 8), namely node and edge markings, variable substitution (including process locations) and group substitution. Examples of such properties are given in Fig. 6.

3.3 Experiments

Experiments were conducted on a 3.5 GHz Intel Core i7 processor (dual core) laptop with 16 GB of memory and running MacOS. Results for a selection of

```
MODULE Properties
$SafeCollaboration \triangleq$   generic property (1)
  $\Box(\forall\, e \in Edge : CatE[e] \in SeqFlowType \Rightarrow edgemarks[e] \leq 1)$

LOCAL $SoundProcessInt(p) \triangleq$
  $\wedge$  $\forall\, e \in Edge : source[e] \in ContainRel[p]$
                  $\wedge\ target[e] \in ContainRel[p] \Rightarrow edgemarks[e] = 0$
  $\wedge$  $\forall\, n \in ContainRel[p]$ :
            $\vee\ CatN[n] \in StartEventType$
            $\vee\ nodemarks[n] = 0$
            $\vee\ nodemarks[n] = 1 \wedge CatN[n] \in EndEventType$

$SoundProcess(p) \triangleq \Diamond SoundProcessInt(p)$   generic property (2)

$SoundCollaboration \triangleq$
  $\Diamond\Box(\wedge \forall\, n \in Node : CatN[n] = Process \Rightarrow SoundProcessInt(n)$
        $\wedge \forall\, e \in Edge : CatE[e] \in MessageFlowType \Rightarrow edgemarks[e] = 0)$

$MessageRelaxedSoundCollaboration \triangleq$   generic property (3)
  $\Diamond\Box(\forall\, n \in Node : CatN[n] = Process \Rightarrow SoundProcessInt(n))$

$User1 \triangleq$   user specific property (4): always end at the base
  $\Diamond\Box(nodemarks[\text{"PlanterEndInBase"}] \geq 1)$

$User2 \triangleq$   user specific property (5): always end with no place to plant
  $\Diamond\Box(subs[\text{"toPlant"}] = \{\})$
```

Fig. 6. Generic and model-specific properties in TLA$^+$.

models from our model database (available from [12], under models/) are presented in Table 2. The first column is the reference of the example in our model database. The characteristics of a model are: number of participants, number of nodes (incl. gateways), number of flow edges (sequence or message flows). When there is communication, we use a bag communication asynchronous model [8]. The results of the verification then follow. First, data on the resulting transition system are given: number of states, number of transitions, and depth (length of the longest sequence of transitions that the model checker had to explore). For each of the correctness properties presented above, we indicate if the model satisfies it. Lastly, the accumulated time for the verification of the properties is given (the time to transform BPMN models to TLA$^+$ transformation is negligible). We selected these five properties since they include BPMN specific generic ones [5], model-specific ones, and cover the safety vs liveness spectrum.

The first model, s004, is a single crop planting process with a loop (find a zone to be planted, move to it, plant it, and mark it to be sprayed) that exits when there are no such zones. Model s006 is our example given in Fig. 3. In this model (and other ones), the id of the end at base TEE is PlanterEndInBase and is used in verification (see property $User1$ in Fig. 6). Model s007 is a variant of model s006 where the context is updated in the planter rather than in the controller. Further, the planter uses the here space formula in the update action rather than znc. Model s008 is the full model (with processes inspired by s007),

Table 2. Experimental results (the space structure is the one in Fig. 2b).

Ref.	Characteristics			Com.	LTS size			Validity					Total
	Proc.	Nodes (gw.)	SF/MF	model	States	Trans.	Depth	(1)	(2)	(3)	(4)	(5)	time
s004	1	6 (1)	5/0	None	34	34	34	✓	✓	✓	✓	✓	<1 s
s006	2	32 (7)	30/5	Bag	276	437	118	✓	✓	✓	×	✓	2 s
s007	2	31 (7)	29/5	Bag	67	104	31	✓	✓	✓	✓	✓	1 s
s008	3	67 (16)	74/10	Bag	4898	16263	53	✓	✓	✓	✓	✓	8 m 31 s
s009	2	33 (7)	31/5	Bag	296	472	123	✓	✓	✓	×	✓	3 s

with both robots and two parallel robot-specific sub-processes in the controller. Finally, s009 is a variant of model s006 where the system does not stop upon an unreachable zone to plant, rather the zone is moved to a notTreated group.

Experiments show that our tool supports the verification of BPMN models with the different extensions we propose and is rather fast, even in presence of loops. Verification takes less than a few seconds for most models. It should be noted that property (1) being a safety property, it requires the whole state space to be constructed when the model satisfies the formula. With model s008 we can see the impact of non-determinism in the model, as we have both intra-process concurrency (two parallel sub-processes in the controller), inter-process concurrency (three peers with message communication), and existential quantification over the movements of the processes (planter and sprayer can move to several places each time). Still, verification is achieved in reasonable time.

4 Related Work

Context awareness and context modelling is widely studied in different domains such as Web systems, mobile applications, ubiquitous computing, and business process engineering [15]. Most approaches that support context awareness for Business Process Management (BPM) provide a categorization of contextual information [2,13–15]. Location is there stressed as an essential element.

A classification of contextual information into categories, that include geographical ones, is proposed in [2], with each category referring to multiple context values. The location model, in our work, is represented by the space structure. Many information reported in [1] were useful to identify its elements. Two kinds of coordinate systems, geometric and symbolic coordinates, are mentioned in [1]. Our space structure relies on the latter, with basic locations. According to [1], for symbolic coordinates and in order to allow spatial reasoning, explicit information about the spatial relations between pairs of symbolic coordinates has to be provided. This led in our approach to two instances of such relations: a static one, the $\rightarrow$ relation, and a dynamic one, the subs relation in configurations.

A survey on spatial models of context information is given in [1], yet without an appropriate model for locations in BPM. The only context modeling approach for BPM we are aware of is [15]. It includes location as a context entity with two attributes, zipcode and city. This is subsumed by our approach as these attributes, and their relations, can be represented using the $\rightarrow$ and subs relations.

Our space structure could be implemented with the data-related constructs of BPMN. Still, this would be cumbersome in comparison to a domain-specific extension like ours. This would also harm the separate reuse of contexts and process models which is made possible by our integration. While there are tools to animate BPMN models with data [4], there is a lack of tools for their verification. Choosing an intermediary expressiveness level, with just the data required for spatial concerns, made verification amenable in our approach. The approach presented in [3] presents interesting perspectives in terms of verification. This approach operates with to distinct databases: a static one (catalog) and an evolving one (repository). Since we deal with environments evolving as a result of the action of processes, we would have to study whether group locations could be treated as part of the catalog and the subs relation as part of the repository.

5 Conclusion and Perspectives

We have proposed a BPMN extension for spatial concerns. Its principles are indeed applicable to other notations such as UML activity diagrams. This widens the use of this family of notations to systems where mobile processes act on their environment. Through the definition of a first-order semantics, its implementation in TLA$^+$, and an automated model transformation, we made it possible to check both the usual process properties (safety, soundness) and context-specific properties. When properties are not satisfied by models, our tools generate counter-examples that are animated on the BPMN models. Our tools and model database are open source and available online at [12].

A first perspective of this work is to consider that the actions of processes can modify the possible moves in the spatial context. We already partially support this through the definition and updates of group locations, but modifying the $\rightarrow$ relation itself would add more expressiveness. By now, processes share the variables set by the evaluation of conditional flows. An extension would be to have also a local perspective of variables, and use message payloads to share them. Taking into account that the environment can evolve by itself (or under the action of processes not being modelled) is also an interesting perspective. This could be achieved, for example, with a specification of possible evolutions to be taken into account at verification time in conjunction with the properties of interest. Indeed, while we use here LTL (with TLA+ extensions to address model-specific properties), we could also study the use of LTL extensions to spatial concerns [7,10]. In this work, we base on the BPMN execution semantics given in [8] with the non-deterministic treatment for time-related elements given there. Another perspective concerns taking into account the second semantics for time given in [8], and study the interplay between time and space. A last perspective is to integrate an optimization aspect to the models, enabling processes to choose their path based on more complex, optimizing strategies, rather than on space formulas only.

Acknowledgements. The authors would like to thank Markus Alexander Kuppe and Philippe Quéinnec for their help with TLA$^+$ and the TLA$^+$ Toolbox.

References

1. Bettini, C., et al.: A survey of context modelling and reasoning techniques. Pervasive Mob. Comput. **6**(2), 161–180 (2010)
2. Born, M., Kirchner, J., Müller., J.P.: Context-driven business process modelling. In: Proceedings of the Joint Workshop on Advanced Technologies and Techniques for Enterprise Information Systems (2009)
3. Calvanese, D., Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: Formal modeling and SMT-based parameterized verification of data-aware BPMN. In: Proceedings of the International Conference on Business Process Management (2019)
4. Corradini, F., Muzi, C., Re, B., Rossi, L., Tiezzi, F.: MIDA: multiple instances and data animator. In: Proceedings of the International Conference on Business Process Management (2018)
5. Corradini, F., et al.: A classification of BPMN collaborations based on safeness and soundness notions. In: Proceedings of the International Workshop on Expressiveness in Concurrency and of the Workshop on Structural Operational Semantics (2018)
6. Dey, A.K.: Understanding and using context. Pers. Ubiquit. Comput. **5**(1), 4–7 (2001)
7. Haghighi, I., Jones, A., Kong, Z., Bartocci, E., Grosu, R., Belta, C.: Spatel: a novel spatial-temporal logic and its applications to networked systems. In: Proceedings of the International Conference on Hybrid Systems: Computation and Control (2015)
8. Houhou, S., Baarir, S., Poizat, P., Quéinnec, P., Kahloul, L.: A first-order logic verification framework for communication-parametric and time-aware BPMN collaboration. Inf. Syst. 101765 (2021, in press). https://doi.org/10.1016/j.is.2021.101765
9. Lamport, L.: Specifying Systems. The TLA+ Language and Tools for Hardware and Software Engineers, Addison-Wesley, Boston (2002)
10. Nenzi, L., Bortolussi, L., Ciancia, V., Loreti, M., Massink, M.: Qualitative and quantitative monitoring of spatio-temporal properties with SSTL. Log. Methods Comput. Sci. **14**(4), 1–38 (2018)
11. OMG Group: Business process modeling notation (2013). http://www.omg.org/spec/BPMN/2.0.2
12. Poizat, P., et al.: fbpmn repository (2021). https://github.com/pascalpoizat/fbpmn
13. Rosemann, M., Recker, J.: Context-aware process design exploring the extrinsic drivers for process flexibility. In: Proceedings of the CAISE 2006 Workshop on Business Process Modelling, Development, and Support (2006)
14. Saidani, O., Nurcan, S.: Towards context aware business process modelling. In: Proceedings of the CAiSE 2007 Workshop on Business Process Modeling, Development, and Support (2007)
15. Saidani, O., Rolland, C., Nurcan, S.: Towards a generic context model for BPM. In: Proceedings of the Annual Hawaii International Conference on System Sciences (2015)
16. Schilit, B.N., Theimer, M.M.: Disseminating active map information to mobile hosts. IEEE Netw. **8**(5), 22–32 (1994)