Customizing BPMN Diagrams using Timelines

Carlo Combi[1], Barbara Oliboni[1], and Pietro Sala[1]

[1]Department of Computer Science, University of Verona, Italy
{carlo.combi, barbara.oliboni, pietro.sala}@univr.it

Abstract

BPMN (Business Process Model and Notation) is widely used for modeling Business Processes by using BPMN Diagrams, but lacks in some aspects. Representing execution-dependent and time-dependent decisions in BPMN Diagrams may be a daunting challenge [10]. In many cases such constraints are omitted in order to preserve the simplicity and the readability of the process model. However, for purposes such as compliance checking, process mining, and verification, formalizing such constraints could be very useful. In this paper, we propose a novel approach for annotating BPMN Diagrams with *Temporal Synchronization Rules* borrowed from the timeline-based planning field. We discuss the expressivity of the proposed approach and show that it is able to capture a lot of complex temporal-related constraints without affecting the structure of the BPMN Diagram. Finally, we provide a mapping from annotated BPMN Diagrams to timeline-based planning problems that allows one to take advantage of the last twenty years of theoretical and practical developments in the field.

1 Introduction

Nowadays Process-Aware Information Systems (PAISs) have become the cornerstone for organizing activities in most realities ranging from large private companies (operating in logistics, manufactoring, avionics, etc.) to healthcare institutions. Business Process Management deals with many important aspects such as analysis, modeling, execution, and monitoring of Business Processes [20].

In this context, BPMN (Business Process Model and Notation) [26] is the standard for representing and managing business processes, but it lacks in some aspects such as the specification of (i) temporal constraints [10, 27], (ii) resources availability [11], and (iii) external data affecting decisions [29].

As pointed out by many applications, time-awareness is a crucial property of business processes in most of the domains and especially in the healthcare one [19, 27]. However, BPMN does not directly allow the specification of time constraints in process diagrams, despite the fact that they affect the real process flow in many aspects such as choices to be made at given decision points, event handling, task durations, resource allocation and so on. This limitation has been considered in the literature in different ways. A possible approach is to extend BPMN with constructs borrowed from workflows and simple temporal networks fields [10, 27]. Another approach consists of translating BPMN diagrams into logical or automata-based formalisms [12, 21] and then expressing constraints by means of the considered formalisms.

Moreover, BPMN does not allow the representation of resource availability and external data affecting decisions, even if these aspects are crucial in managing process execution and outcome. A further issue to be considered is that resources and data values change over time. As an example, in the healthcare domain, resource availability with respect to blood analysis may be affected by the time of the day (i.e., morning, afternoon, evening, and night) and the current load of the lab (i.e., the number of analysis in execution). Time of the day and current load may influence the whole time required for getting results of blood tests. An example taking into account external data affecting decisions is related to shifts in the systolic blood pressure values of a patient undergoing a surgical procedure. Significant differences in pressure values in last 5 hours may force the anaesthetist to administer a local sedation in place of a total one for safety reasons.

For the sake of clarity and conciseness of BPMN Diagrams, often the formal specification of these aspects are intentionally neglected and left to the following implementation with specific software tools.

In this paper, we propose an approach, residing in between the two aforementioned ones, that allows the annotation of BPMN diagrams based on *Temporal Synchronization Rules* of *Timeline-based planning* [25]. We also show that this simple language may naturally express the specification of (i) temporal constraints, (ii) resource availability, and (iii) external data affecting decisions. Moreover, the proposed approach allows one to constrain the execution of the process (e.g., a decision in an exclusive-gateway) according to the aforementioned specifications. Then, another contribution of this work is a complete mapping of our timeline-annotated BPMN Diagrams into a timeline-based planning problem, that is, given a set of state variables and a set of synchronization rules on them find a consistent execution where all the synchronization rules are satisfied [25]. The traslation step suffices for our verification purpose, since various tools for satisfiability of timeline-based planning have been proposed in the last decade [2, 5, 6].

Advantages of our proposal are manyfold:

1. The proposed approach allows us to express complex temporal constraints even if they involve some external data or resources.

2. The temporal behaviour of data and resources may be regulated with

the same machinery (i.e., state variables).

3. Our approach favours composability. As a matter of fact, resources/data may be updated/removed/inserted, as well as temporal constraints on the execution of the business process, by simply modifying the relative temporal synchronization rules/state variables.

4. The process diagram is not affected at all and it may be seen through a layered perspective: (a) at the highest level, the original BPMN Diagram provides a general idea of how activities are organized; (b) at an intermediate level, temporal synchronization rules, possibly involving one or more external entities, detail how the execution is temporally constrained and how some decision points are affected by (the temporal evolution of) data/resources and/or by some previous temporal behavior of the process; (c) finally, at the lowest level, the state variables regulate the evolutions of the involved data/resources.

Power and generality of this approach come at the price that the definition of a set of temporal constraints in the form of temporal synchronization rules and state variables that could be inconsistent (i.e., every possible legit execution of the diagram combined with every possible consistent evolution of data/resources violates at least one temporal synchronization rule).

In this paper, we will focus only on structured BPMN Diagrams, and thus from now on we will call them just diagrams. A diagram is said to be well-structured if every node with multiple outgoing edges, i.e., a split node, has a corresponding node with multiple incoming edges, i.e., a join node, such that the set of nodes delimited by the split and the join nodes form a Single-Entry-Single-Exit (SESE) region, and these regions within the process are properly nested [14, 18]. In this way a SESE region is any area within a process delimited by a single entry edge and a single exit edge.

The paper is organized as follows. Sec. 2 gives an overall description of the proposed approach. Sec. 3 provides an example of a real-world process in the healthcare domain, which features non-trivial temporal constraints. Sec. 4 recalls the basic concepts and notation of timelines and timeline-based planning, together with some recent results in the field. Sec. 5 shows some meaningful temporal constraints that may be enforced by means of timeline annotations in BPMN Diagrams in a straightforward way, without compromising the overall readability of the diagram. Sec. 6 describes how the proposed approach allows the specification of constraints involving data, resources, and decisions. Sec. 7 summarizes the contribution of the paper and sketches some lines for future work.

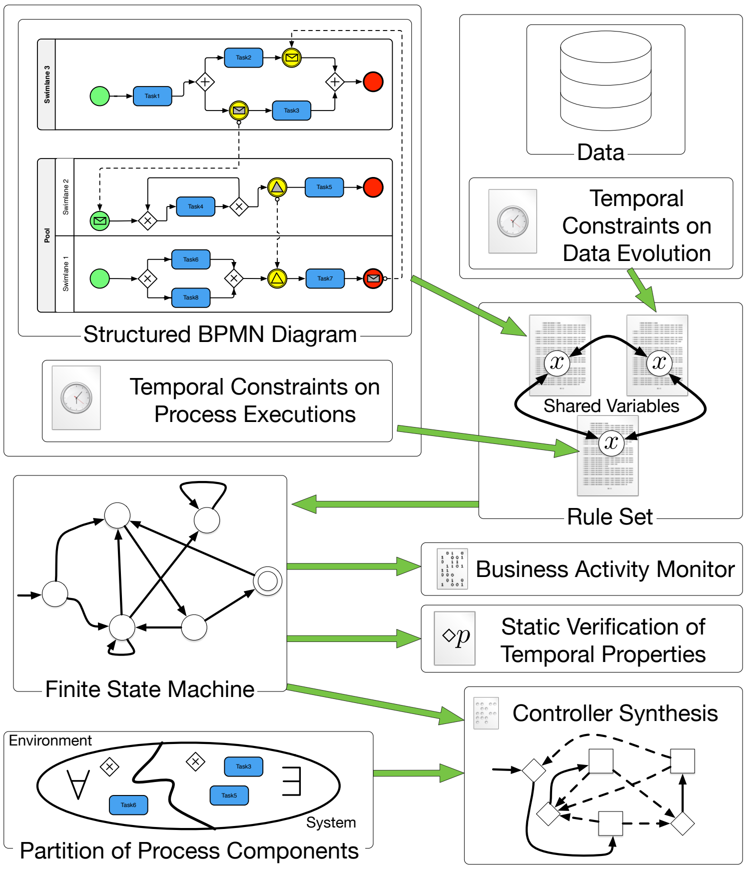

Figure 1: A pipeline for integrating timeline-based planning and BPMN diagrams.

2 Enriching BPMN with Timelines: the Big Picture

In this section we give an overview of the proposed approach which is graphically summarize in Fig. 1. Ideas and motivations behind our proposal are the following. BPMN Diagrams are often used for modelling businesss processes by considering time-critical, resource-critical, and data-critical situations and usually they are underspecified w.r.t. such requirements for preserving their readability and conciseness. In this simplified form they cannot be directly translated into a suitable algorithm for controlling the whole process at runtime and/or for performing qualitative/quantitative static analysis. On the other hand, forcing the representation of such requirements by enriching the diagram will compromise the readability of the diagram itself.

In order to overcome such trade-off, our proposal consists of keeping the original diagram and annotating it by using a set of constraints, namely *temporal synchronization rules*, borrowed from the timeline-based planning domain. As we will detail in Sec. 5, such rules are able to express in a concise and clear way temporal constraints that would otherwise be captured by a complex combination of throw/catch events and event-based gateways [10]. In Appendix B, we will provide a way to translate structured BPMN processes into a set of rules representing all and only the possible correct executions of the process. Such mapping is crucial because, as shown in Fig. 1, it allows the representation of both requirements and process as a set of rules.

In Fig. 1, we suppose to have a process that makes use of some data, and constraints on such data must be taken into account. For instance, let us assume that the diagram represents a medical guideline in which the decision on the exclusive gateway is driven by the value of the patient blood pressure. It is straightforward to see that such value cannot increase/decrease too fast in a short amount of time and it would be desirable to force such constraint in order to prune irrealistic behaviors of the process in subsequent analysis. In this paper, we assume that constraints on data may be captured by a suitable set of temporal synchronization rules. As shown in Fig. 1, as a first step, the diagram, temporal constraints, and data constraints are translated into sets of rules in an independent manner. The union of such obtained rules (the Rule Set in Fig. 1) represents the whole description of the considered process. As mentioned before, the translations are pairwise independent but, as we will observe in Sec. 6, rules in different sets may "communicate" via shared variables. For instance, rules representing the diagram may involve the variable representing the pressure, whose behaviour is encoded by other rules coming from data constraints. Another example may be represented by the fact that a given temporal constraint imposes that the execution of two specific tasks must be non-overlapping (since they use the same shared resource), no matter how they are arranged in the diagram (e.g., they may appear in parallel branches). It is easy to see that such approach fosters modularity in the design of every component. As a matter of fact, we may change constraints on the behavior of data, without affecting the diagram, or we may change the diagram without impacting on related temporal constraints.

As depicted in Fig. 1, the whole set of rules is translated into a Finite State Machine (FSM), whose language represents all the possible correct executions of the considered diagram w.r.t. to temporal/data constraints. The FSM may be used for performing a plethora of process-related analysis. In Fig. 1, we just provide three of them. (i) FSM may be translated into an algorithm that may be used at runtime for monitoring the correct execution of the process by means of alerts/execeptions pointing out the violation of a given constraint [15]. (ii) On the FSM we may perform static verification of qualitative/quantitative properties, expressed in temporal logics such as LTL or CTL [17], by using one of the many well-established tools on the market [8, 16]. (iii) Supposing to be in a scenario where some process elements are under the control of the environment (e.g., medical guidelines). Then, by means of the FSM, we may synthesize, if it exists, a controller that "drives" the system-controlled elements (i.e., the process elements which are not controlled by the environment) in a way that the correct termination of the process is ensured, no matter how the environment behaves on its set of elements [24, 28].

3 A motivating example

In this section, we introduce a clinical process model and describe some time-related decisions and constrains that can be considered. The Business Process model, represented in Fig. 2 as a BPMN Diagram, is a process for the treatment of Catheter-Related Bloodstream Infections (CR-BSIs). Vascular catheters are vital for treating ill patients in critical situations. Their main drawback is represented by the concrete possibility of a patogens colonization of their injection site. This may lead patients to develop severe bloodstream infections.

Clinical guidelines for preventing such infections have been proposed and applied, most of them usually rely on temporal constraints for their applicability [4]. The BPMN diagram in Fig. 2 shows the process for detecting and treating CR-BSIs according to the well-known Infectious Diseases Society of America (IDSA) practice guideline [22]. The guideline includes blood and/or catheter cultures activities for supporting the diagnosis of CR-BSI. In particular, clinicians first draw *simultaneously* two blood samples to be cultured, one from the catheter suspected to be the source of the infection and, the other, from a peripheral vein. We call the first sample *LS* (local

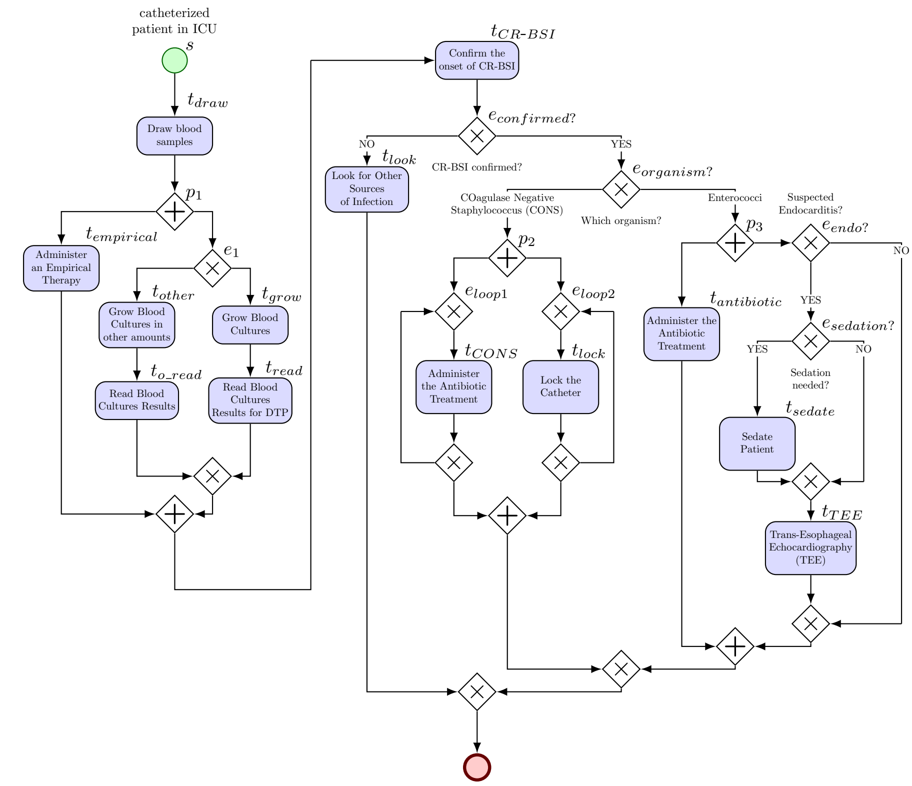

Figure 2: A BPMN Diagram representing CR-BSIs treatment.

sample) and the second one PS (peripheral sample), respectively. These operations are included in the first process activity of Fig. 2, i.e. *Draw blood samples*. The considered activity takes a t_{draw} time to be completed.

After the first activity, physicians *Administer an empirical therapy* to the patient until the diagnosis of CR-BSI is confirmed. Among the criteria for confirming or not a CR-BSI, we considered the Differential Time to Positivity (DTP), which measures the difference between the time when LS becomes positive w.r.t. a certain micro-organism, and the time when PS becomes positive for the same micro-organism. If such difference exceeds a certain threshold (DTP), then the CR-BSI is confirmed.

In the process of Fig. 2, we considered only two of the possible micro-organisms that may be detected in a CR-BSI infection: Coagulase-negative Staphylococci and Enterococcus spp.

- In case of *Coagulase-negative Staphylococci (CONS)*, patient is treated with antibiotic or heparin lock therapy. Such therapy consists of alternating between catheter locks and an antimicrobial therapy. In general, such phases have equal duration in order to prevent clot formations. These activities are represented in Fig. 2 by means of the process region related to gateway *p2*, composed of *Administer Antibiotic Treatment* and *Lock Catheter* activities.

- In case *Enterococcus spp*, patient is treated by administering Vancomycin. Unfortunately this case is often associated with endocarditis. This means that physicians may choose to perform a Trans-Esophageal Echocardiography (TEE) for detecting the issue. TEE must be performed not before five and up to seven days from the time when CR-BSI has been confirmed. These activities are represented in Fig. 2 by means of the process region related to gateway *p3*.

Summing up, even in this over-simplified representation of a real-world clinical scenario, we need to specify time-related constraints, which cannot be captured by using BPMN without compromising the process model clarity.

Examples of these time-related constraints are:

- Duration-Induced-Decision (*DID*). Durations and interleaving of given events/tasks preceding a decision point (i.e., an exclusive gateway), determines the choice to be made, and thus the path to follow. In process of Fig. 2, time durations *LS* and *PS* and their related DTP determine which branch of *CR-BSI confirmed?* will be taken.
- Disjoint-Parallel-Tasks (*DPT*). In this case, we consider tasks which may be executed without a given order, but their execution needs to be disjoint for some reasons (e.g., the preemption of a mutually exclusive resource). In the treatment of CONS, *Administer Antibiotic Treatement* and *Lock Catheter* must be executed in a non-overlapping way. Moreover, since in Fig. 2, both activities belong to a loop, they may be executed multiple times.
- Relative-Time-Constraint (*RTC*). Time durations of two given tasks, or the difference between their endpoints are constrained by specified bounds. In process of Fig. 2, the difference between the beginning of the *TEE* activity, and the end of the *CR-BSI* activity must be between five and seven days.

4 A formal account of Timelines

In this section we introduce the basic concepts timelines and of timelines-based planning [25]. Appendix A introduces an informal explanation, together with a little example, of how the whole timelines-based machinery works. In the following, we use the notation introduced in [7]. We start by introducing the notion of *state variable*.

Definition 1. ***(state variable)*** *A state variable sv is a triple* $sv = (\mathbf{V}_{sv}, \mathbf{\Delta}_{sv}, \mathbf{D}_{sv})$ *where:*

- $\mathbf{V}_{sv}$ *is the finite domain of the state variable sv;*
- $\mathbf{\Delta}_{sv} : \mathbf{V}_{sv} \rightarrow 2^{\mathbf{V}_{sv}}$ *is the transition function, which maps each value* $v \in \mathbf{V}_{sv}$ *to the set of values that may be taken by sv immediately after sv has taken value v;*

- $\mathbf{D}_{sv} : \mathbf{V}_{sv} \to \mathbb{N} \times \mathbb{N} \cup \{+\infty\}$ *is a function that maps each* $v \in \mathbf{V}_{sv}$ *to an interval, i.e., a pair of values* $[d^{sv=v}_{\min}, d^{sv=v}_{\max}]$ *with* $0 < d^{sv=v}_{\min} \leq d^{sv=v}_{\max}$*, which represent respectively the minimum and the maximum duration of an interval over which sv takes value v.*

Given a state variable sv a *timeline for sv* is a sequence $\mathbf{T}_{sv}$ of pairs called *tokens* which respect functions $\mathbf{\Delta}_{sv}$ and $\mathbf{D}_{sv}$. Formally:

Definition 2. ***(token)*** *A token for a state variable* $sv = (\mathbf{V}_{sv}, \mathbf{\Delta}_{sv}, \mathbf{D}_{sv})$ *is a tuple* $\tau = \langle v, d \rangle$ *where* $v \in \mathbf{V}_{sv}$ *and* $d \in \mathbf{D}_{sv}(v)$*.*

Definition 3. ***(timeline)*** *A timeline for a state variable* $sv = (\mathbf{V}_{sv}, \mathbf{\Delta}_{sv}, \mathbf{D}_{sv})$ *is a finite sequence* $\mathbf{T}_{sv} = \langle \tau_1, \ldots \tau_k \rangle$ *of tokens for sv such that for every* $1 \leq i < k$ *we have* $v_{i+1} \in \mathbf{\Delta}_{sv}(v_i)$*.*

Given a token $\tau = \langle v, d \rangle$ we denote with $value(\tau)$ its value (i.e., $value(\tau) = v$). Notice that the value of sv in two consecutive tokens within a timeline do not need to be different, it depends on how $\mathbf{\Delta}_{sv}$ is defined. Given a timeline $\mathbf{T}_{sv}$ we denote with $|\mathbf{T}_{sv}|$ its length. Moreover we will use an array-like notation for specific tokens in the sequence, formally, if $\mathbf{T}_{sv} = \langle \tau_1, \ldots \tau_k \rangle$ we have $\mathbf{T}_{sv}[i] = \tau_i$ for every $1 \leq i \leq k$. In a timeline $\mathbf{T}_{sv} = \langle \tau_1, \ldots \tau_k \rangle$ for sv for every $1 \leq i \leq k$ we define $\mathbf{s}\_time(\mathbf{T}_{sv}, i)$ as $\mathbf{s}\_time(\mathbf{T}_{sv}, i) = \sum_{1 \leq j < i} d_j$ and $\mathbf{e}\_time(\mathbf{T}_{sv}, i)$ as $\mathbf{e}\_time(\mathbf{T}_{sv}, i) = \sum_{1 \leq j \leq i} d_j$. In the following we will often refer to specific sets of timelines instead of single ones. To this purpose, given a set of timelines $\Gamma = \{\mathbf{T}_{sv_1}, \ldots, \mathbf{T}_{sv_n}\}$, we will say that Γ is *repetition-free* if and only if $sv_i \neq sv_j$ for every $1 \leq i \neq j \leq n$. From now on we will assume every set of timelines to be repetition-free. Synchronization among timelines in the same set is given by means of a set of *Temporal Synchronization Rules*, TS-RULES for short. TS-RULES relate tokens, possibly belonging to different timelines, through temporal relations among intervals called *atoms*. Let $\Sigma = \{x, y, z, \ldots\}$ a set of *token names* (i.e., variables ranging over tokens):

Definition 4. ***(atom)*** *An atom is a clause of the form* $x \leq^{\circ,\bullet}_{I} y$ *where* $\circ, \bullet \in \{\mathbf{s}, \mathbf{e}\}$ *and* $I \in \{[l, u], [l, +\infty) : l, u \in \mathbb{N}, l \leq u\}$*.*

In the above definition $\mathbf{s}$ (resp., $\mathbf{e}$) refers to the start (resp., end) time of tokens x and/or y. By means of *conjuctions* of atoms we may express all the possible Allen's interval relations [1] between two tokens, and some disjunctions of them. In particular, we will use the shorthands reported in Fig. 3. Tokens appear in conjunctions which are existentially closed for all but one distinguished variable.

Definition 5. ***(existential*** x***-free conjunction)*** *Given a token name* x *an existential* x*-free conjunction is a formula* $\mathcal{E}$ *of the form*

$$\mathcal{E} = \exists x_1[sv_1 = v_1] \ldots \exists x_h[sv_h = v_h](A_1 \wedge \ldots \wedge A_m)$$

shorthand	meaning	translation
$x\langle M\rangle y$	x **meets** y	$x \leq^{e,s}_{[0,0]} y$
$x\langle B\rangle y$	x **begins** y	$x \leq^{s,s}_{[0,0]} y \wedge x \leq^{e,e}_{[1,+\infty)} y$
$x\langle D\rangle y$	x **during** y	$y \leq^{s,s}_{[1,+\infty)} x \wedge x \leq^{e,e}_{[1,+\infty)} y$
$x\langle F\rangle y$	x **finishes** y	$y \leq^{s,s}_{[1,+\infty)} x \wedge x \leq^{e,e}_{[0,0]} y$
$x\langle O\rangle y$	x **overlaps** y	$x \leq^{s,s}_{[1,+\infty)} y \wedge x \leq^{e,e}_{[1,+\infty)} y \wedge y \leq^{s,e}_{[1,+\infty)} x$
$x \subset_{BD} y$	(x **begins** y) $\vee$ (x **during** y)	$y \leq^{s,s}_{[0,+\infty)} x \wedge x \leq^{e,e}_{[1,+\infty)} y$
$x \subseteq y$	(x **begins** y) $\vee$ (x **finishes** y) $\vee$(x **during** y) $\vee$ ($x = y$)	$y \leq^{s,s}_{[0,+\infty)} x \wedge x \leq^{e,e}_{[0,+\infty)} y$
$x \cap_{BMO} y$	(x **begins** y) $\vee$ (x **meets** y) $\vee$(x **overlaps** y)	$x \leq^{s,s}_{[0,+\infty)} y \wedge y \leq^{s,e}_{[0,+\infty)} x \wedge x \leq^{e,e}_{[1,+\infty)} y$
$x = y$	$x = y$	$x \leq^{s,s}_{[0,0]} y \wedge x \leq^{e,e}_{[0,0]} y$

Figure 3: A set of useful atoms conjunctions and their interval based interpretation.

where for every $1 \leq i \leq h$ *we have* $v_i \in \mathbf{V}_{sv_i}$ *and* $x_i \neq x$*, moreover for every* $1 \leq j \leq m$ A_j *is an atom of the form* $\overline{x}^1_j \leq^{\circ_j,\bullet_j}_{I_j} \overline{x}^2_j$ *where* $\overline{x}^1_j, \overline{x}^2_j \in \{x_1, \ldots, x_h\} \cup \{x\}$*.*

Informally, in an existential x-free conjunction, a variable in the atom is existentially closed or equal to the unique free variable x. Moreover, we will say that an existential x-free conjunction $\mathcal{E}$ is an *existentially closed conjunction* if and only if for every $1 \leq j \leq m$, A_j is an atom of the form $\overline{x}^1_j \leq^{\circ_j,\bullet_j}_{I_j} \overline{x}^2_j$ where $\overline{x}^1_j, \overline{x}^2_j \in \{x_1, \ldots, x_h\}$ (i.e., $\mathcal{E}$ does not feature any free-variable). From now on we will treat the case of x-free conjunction which are not existentially closed. Existentially closed conjunctions may be seen as a special case of x-free ones so we will omit them for the sake of brevity. Moreover, given an x-free conjunction $\mathcal{E} = \exists x_1[sv_1 = v_1] \ldots \exists x_h[sv_h = v_h](A_1 \wedge \ldots \wedge A_m)$ we define $SVar(\mathcal{E}) = \{sv_1, \ldots, sv_h\}$ as the set of state variables in its existential preamble. Analogously, we define $TNames(\mathcal{E}) = \{x_1, \ldots, x_h, x_{h+1}\}$ assuming without loss of generality that x_{h+1} is the free variable x.

Since the token names $TNames(\mathcal{E})$ are exactly $h+1$ (all the existentially quantified ones plus the free one x), we may have that $SVar(\mathcal{E}) \leq h$ because it is absolutely legit that two distinct token names are bound to the same state variable.

Semantics for x-free conjunctions $\mathcal{E} = \exists x_1[sv_1 = v_1] \ldots \exists x_h[sv_h = v_h](A_1 \wedge \ldots \wedge A_m)$ is given in terms of a set of timelines $\Gamma = \{\mathbf{T}_{\overline{sv}_1}, \ldots, \mathbf{T}_{\overline{sv}_n}\}$ such that $SVar(\mathcal{E}) \subseteq \{\overline{sv}_1, \ldots, \overline{sv}_n\}$, a state variable sv_{h+1} in $\{\overline{sv}_1, \ldots, \overline{sv}_n\}$ (i.e, $\mathbf{T}_{sv_{h+1}}$ is the timeline that will be associated to x), and a function $f : TNames(\mathcal{E}) \to \mathbb{N}$. In such setting we will have that $\Gamma, sv_{h+1}, f \models \mathcal{E}$ if and only if the following conditions hold:

- for every $1 \leq i \leq h+1$ we have $|\mathbf{T}_{sv_i}| \geq f(x_i)$;

- for every $1 \leq i \leq h$ we have $value(\mathbf{T}_{sv_i}[f(x_i)]) = v_i$
- for every $1 \leq j \leq m$ let $A_j = x_{i_j} \leq^{\circ_j,\bullet_j}_{[l_j,u_j]} x_{i'_j}$ (resp., $A_j = x_{i_j} \leq^{\circ_j,\bullet_j}_{[l_j,+\infty)} x_{i'_j}$) for some $1 \leq i_j, i'_j \leq h+1$, then

$$l_j \leq \bullet_j\_time(\mathbf{T}_{i'_j}, f(x_{i'_j})) - \circ_j\_time(\mathbf{T}_{i_j}, f(x_{i_j})) \leq u_j$$

(resp., $l_j \leq \bullet_j\_time(\mathbf{T}_{i'_j}, f(x_{i'_j})) - \circ_j\_time(\mathbf{T}_{i_j}, f(x_{i_j}))$).

Now we are ready to introduce TS-RULES.

Definition 6. ***(temporal synchronization rule)*** *A temporal synchronization rule $\mathcal{R}$ is a formula which has one of the following two forms:*

- *(trigger rule) $\mathcal{R} = x[sv = v] \rightarrow \mathcal{E}_1 \vee \ldots \vee \mathcal{E}_n$, where for every $1 \leq i \leq n$ we have that $\mathcal{E}_i$ is an existential x-free conjunction;*
- *(triggerless rule) $\mathcal{R} = \mathcal{E}_1 \vee \ldots \vee \mathcal{E}_n$ where for every $1 \leq i \leq n$ we have that $\mathcal{E}_i$ is an existentially closed conjunction.*

For the sake of clarity we will provide only the semantics of trigger rule since triggerless ones are a simplified version of them. Given a trigger rule $\mathcal{R} = x[sv = v] \rightarrow \mathcal{E}_1 \vee \ldots \vee \mathcal{E}_h$ its semantics is given by means of a set of timelines $\Gamma = \{\mathbf{T}_{sv_1}, \ldots, \mathbf{T}_{sv_n}\}$ such that $\{sv_1, \ldots, sv_n\} \supseteq \bigcup_{i=1}^{h} SVar(\mathcal{E}_i) \cup \{sv\}$, in such a case we say that Γ is a *candidate* for $\mathcal{R}$.

Definition 7. ***(semantics of trigger rules)*** *Given a trigger rule $\mathcal{R} = x[sv = v] \rightarrow \mathcal{E}_1 \vee \ldots \vee \mathcal{E}_h$ and a candidate $\Gamma = \{\mathbf{T}_{sv_1}, \ldots, \mathbf{T}_{sv_n}\}$ for it. We that Γ satisfies $\mathcal{R}$, written $\Gamma \models \mathcal{R}$, if and only if for every $1 \leq i \leq |\mathbf{T}_{sv}|$ if $\mathbf{T}_{sv}[i] = v$ then there exist $1 \leq j \leq h$ and a function $f : TNames(\mathcal{E}_j) \rightarrow \mathbb{N}$, for which $f(x) = i$ and $\Gamma, sv, f \models \mathcal{E}_j$.*

The timelines-based planning problem is defined as follows.

Definition 8. ***(timelines-based planning problem)*** *Given a set of TS-RULES $\mathbf{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_p\}$ the timelines-based planning problem, TPP for short, for $\mathbf{R}$ consists of determining whether or not there exists a set of timelines $\Gamma = \{\mathbf{T}_{sv_1}, \ldots, \mathbf{T}_{sv_n}\}$ such that $\Gamma \models \mathcal{R}_i$ for every $1 \leq i \leq p$.*

5 Annotating BPMN Diagrams with Timelines

In this section we describe in more details our approach which consists of annotating BPMN diagrams with temporal synchronization rules. The proposed annotation is able to enrich the description of process execution by maintaining the diagram as simple as possible. In our proposal, we use a synchronization rule based notation which allows us to easily handle temporal constraints represented by means of timelines. We would like to point out that in our approach each set Γ is associated with a possible instance of the process (i.e., Γ may be seen as the whole process log for a given process instance) while state variables together with TS-RULES abstract away from single instances and represent constraints on such instances exactly as the corresponding BPMN process diagram does.

As an example, we consider the BPMN diagram reported in Fig. 2, which is annotated by means of timelines. Appendix B provides a formal mapping from diagrams to timelines-based planning problems. It is easy to prove that such mapping guarantees the existence of a bijection between the solutions of the target planning problem and the correct executions of the related process model. In Fig. 4 we show an instance execution of the considered process. The execution is represented as a set of timelines, one for each BPMN element.

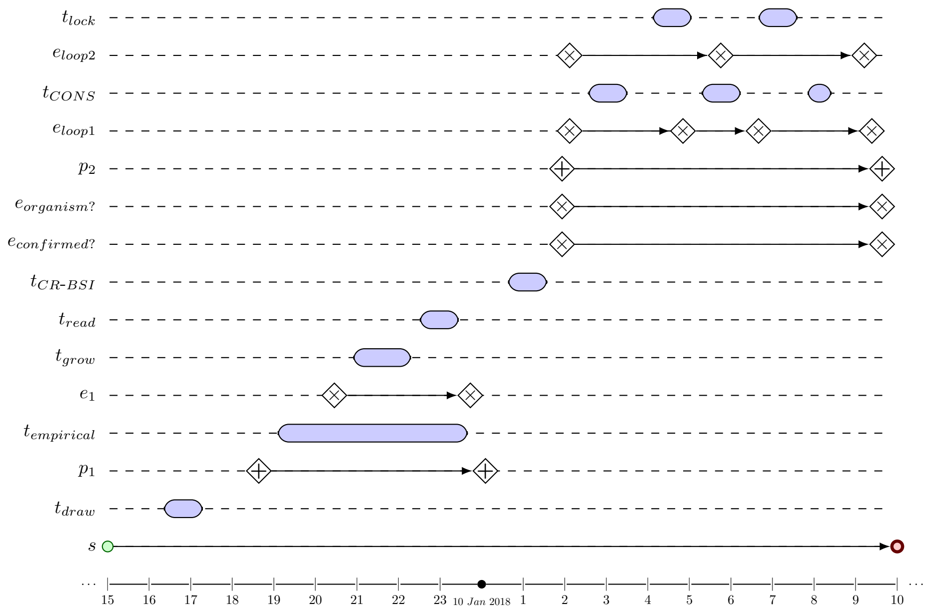

Figure 4: Example of an execution of the business process of Fig. 2, represented as timeline (for the sake of brevity only timelines related to elements involved in the considered execution are shown).

Tokens on timelines may take two values, *active*, denoted by ⊤, and *not active*, denoted by ⊥, this means that each token can be seen as an on/off switch. The meaning of these two values is straightforward, *active* means that the process element is currently executed and its duration is represented by means of the duration of the token, and *not active* means that the process element is not executed in the interval of time corresponding to the token. In Fig. 4, when a token is active, it is represented by using the BPMN notation related to the considered element, otherwise, when the token is not active, it is represented by means of a dashed line. For example, the execution of task *Administer an Empirical Therapy* has a duration of 4 hours and half, as represented in Fig. 4 by using a task-like shape on the $t_{empirical}$ line from 19.00 to 23.30. The execution of task *Administer the Antibiotic Therapy* related to line t_{CONS} is not executed in the 2-hours interval starting at 10 Jan 2018 3:00.

In our proposal, we take advantage from the fact that the BPMN diagram is structured, and associate a timeline to each SESE region. The beginning of an active token represents the entry node (gateway) of the SESE region associated to the timeline, and the ending of such token represents its exit gateway. For instance, in Fig. 4 the two executions related to gateway e_{loop2} are represented by the active tokens [10 Jan 2018 2:00, 10 Jan 2018 5:00] and [10 Jan 2018 5:00, 10 Jan 2018 9:00] on the relative timeline. In Fig. 4, an example of execution of the BPMN Process Diagram of Fig. 2 is reported. In this example tasks and gateway blocks are correctly interleaved.

In Appendix B, we will give more details about the way the described tokens can be properly constrained for representing correct executions of gateways and tasks, and about the way interleaving may be forced by means of suitable synchronization rules.

In the following examples we will assume that the presented scenario is taken from timelines representing correct executions of the considered BPMN process. For example in Fig. 2, a timeline having a token t_{look} which is active before an active token t_{grow} is not allowed since the correct execution of the process requires that the execution of task *Look for Other Sources of Infection* related to t_{look} is after the execution of task *Grow Blood Culture* related to t_{grow}. Before providing the rules for the constraints related to the example of Sec. 3, we introduce a (more human-readable) variation on the syntax for TS-RULES. In our opinion such syntax is more suitable for annotating BPMN diagrams. First, instead of anonymous state variable names like $x, y, \ldots$ we will use the element type associated to the state variable, then we will write something like $task, task', \ldots$ when the state variable is associated to a task, $exclusive, exclusive', \ldots$ when the state variable is associated to a region delimited by an exclusive gateway, and so on. Moreover, we replace *state-variable* = *token-value* in the quantifications with either *element-name* or its overlined version $\overline{element\text{-}name}$ where *element-name* is the subscript of the BPMN element associated to state-variable. We will write *element-name* if *token-value* $= \top$ and $\overline{element\text{-}name}$ if *token-value* $= \bot$, respectively. For instance, rule $x[t_{CONS} = \top] \rightarrow \exists y[t_{lock} = \bot](x \subseteq y)$ turns out to be rule $task[CONS] \rightarrow \exists task'[\overline{lock}](task \subseteq task')$ in the new syntax. The DID, DPT and RTC constraints related to the example of Sec. 3 may be expressed as follows:

- Duration-Induced-Decision (DID):

$$\text{C1)}\quad task[grow] \rightarrow \begin{array}{c} \exists exclusive[organism?] \left(\begin{array}{c} task \leq^{s,e}_{[2\ hours,+\infty)} task \wedge \\ task \leq^{s,s}_{[0,+\infty]} exclusive \end{array} \right) \vee \\ (task \leq^{s,e}_{[0,2\ hours]} task) \end{array}$$

Fig. 5 shows examples of four partial evolutions of timelines $t_{grow}, e_{organism?}$ and t_{look}. These considered scenarios are triggered by the presence of the execution of the task related to t_{grow} (i.e., the rounded rectangle on the bottom dashed line).

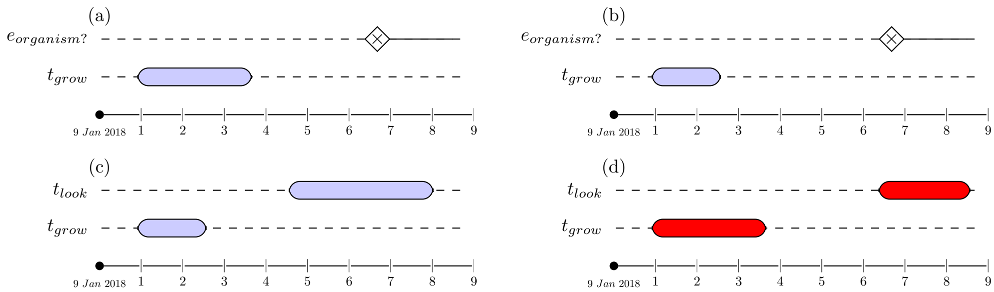

Figure 5: (a), (b), and (c) are examples of executions that fulfill the Duration Induced Decision constraint C1. (d) does not fulfill C1.

Fig. 5.(a) represents the case in which the duration of t_{grow} is more than two hours and thus, according to the specified constraint, the YES branch of $e_{confirmed?}$, and the block $e_{organism?}$, must be executed. In this scenario the first disjunction in C1 is fulfilled. When the duration of t_{grow} is less than 2 hours, either YES branch or NO branch of $e_{confirmed?}$ is executed, as depicted in Fig. 5.(b) and Fig. 5.(c), respectively. In the latter case task related to t_{look} must be executed as correctly depicted in Fig. 5.(c).
Example in Fig. 5.(d) represents a way to violate constraint C1. In this case, the duration of t_{grow} is greater than 2 hours and the branch NO of $e_{confirmed?}$ is taken by executing t_{look}. This situation violates both disjunction of C1.

- Disjoint-Parallel-Tasks (DPT):

$$\text{C2)} \quad task[CONS] \rightarrow \exists task'[\overline{lock}](task \subseteq task')$$

Fig. 6 reports examples of two partial evolutions of t_{lock} and t_{CONS} timelines. The intuition behind rule C2 is that if a token on the timeline t_{CONS} is active, then it is contained in a not active token on the timeline t_{lock}.

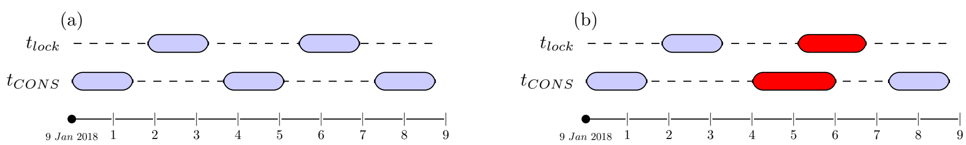

Figure 6: (a) is an example of execution that fulfills the Disjoint-Parallel-Tasks constraint C2. (b) does not fulfill C2.

In Fig. 6.(a) an interleaving of tokens in t_{lock} and t_{CONS} that respects

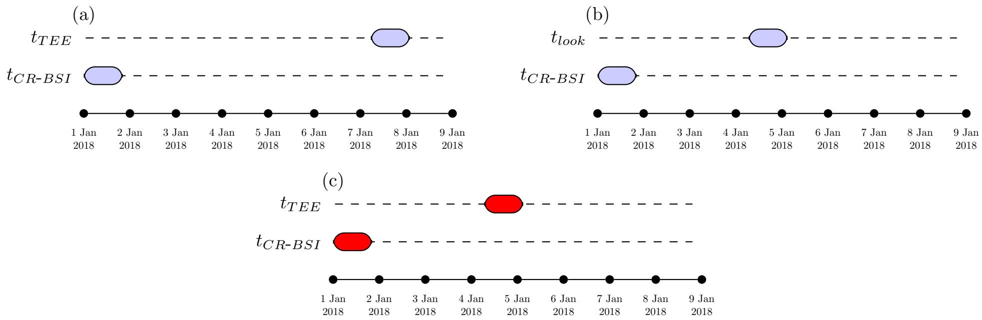

Figure 7: (a) and (b) are examples of executions that fulfill the Relative-Time-Constraint C3. (c) does not fulfill C3.

rule C2 is depicted. In Fig. 6.(b) a scenario that violates rule C2 is reported. In this latter case, token $[9\ Jan\ 2018\ 4{:}00, 9\ Jan\ 2018\ 6{:}00]$ on timeline t_{CONS} contains the overlap of tokens $[9\ Jan\ 2018\ 3{:}30, 9\ Jan\ 2018\ 5{:}00]$ and $[9\ Jan\ 2018\ 5{:}00, 9\ Jan\ 2018\ 6{:}30]$ on t_{lock}, and thus it cannot be contained in any token on timeline t_{lock}.

- Relative-Time-Constraint (RTC):

$$\text{C3)}\ task[CR\text{-}BSI] \rightarrow \begin{array}{l} \exists task'[\overline{TEE}]\exists task''[TEE](task \subseteq task' \\ \wedge task'\langle M\rangle task'' \wedge task \leq^{e,s}_{[5\ days, 7\ days]} task'') \\ \vee \exists task'[\overline{TEE}](task\langle M\rangle task' \wedge task' \leq^{s,e}_{(+\infty,+\infty)} task') \end{array}$$

Fig. 7 shows examples of three partial evolutions involving t_{TEE}, $t_{CR\text{-}BSI}$ and t_{CONS} timelines. Scenario reported in Fig. 7.(a) fulfills rule C3 since an active token on timeline $t_{CR\text{-}BSI}$ is present, and the next active token on t_{TEE} happens after 6 days. Also Fig. 7.(b) represents a scenario satisfying C3. Assuming that timelines respect the correct execution of the process diagram, in this case there is an active token on the timeline t_{look} then, the NO branch of $e_{confirmed?}$ has been chosen, and thus there are no active tokens on timeline t_{TEE}. This means that second conjunction of $C3$ is fulfilled.

Finally, Fig. 7.(c) shows a scenario violating rule C3, since there exists an active token τ on t_{TEE} which happens after an active token τ' on $t_{CR\text{-}BSI}$, but the distance between the end of τ' and the beginning of τ is less than five days.

TS-rulesallow us to capture different kind of constraints For example, the described temporal constraints may be achieved by suitably adding throw/catch events and event-based gateways to the diagram. However, there are two main drawbacks in this approach: (i) enforcing such constraints in the diagram may easily render it unreadable (e.g., see [10] for an example); (ii) modularity is lost forever since some changes in the diagram may change how the constraints are enforced in it.

Moreover, some constraints expressible via TS-rules may be defined by

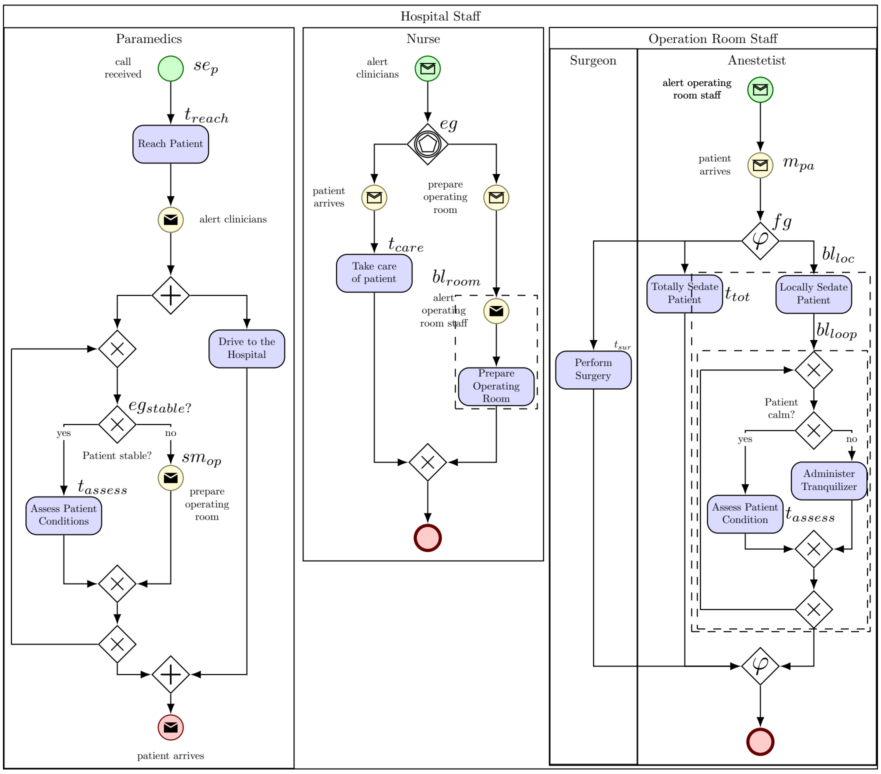

Figure 8: An example of History-driven gateway.

using *Decision Model and Notation* (DMN) [23]. DMN is a standard notation for modeling decisions, and it is complementary to BPMN. DMN is able to specify conditions on the elements that may change the flow of execution (e.g., exclusive gateways). Our approach can capture DMN sematics in a natural way by introducing additional state variables for data affecting the choice (more on that in Sec. 6) and the relative TS-RULES thus providing a way to check consistency properties between the DMN logic and the process. However, if the choices are inherently dependent from the evolution of the data and/or the flow of the process TS-RULES explicit such relation in a more direct and concise way; Finally, TS-RULES are more general since they constrain the flow of execution without the need to be bound to some element in the diagram, for instance, they can force parallel tasks to follow specific patterns as shown by rule C2.

6 Data, Resources, and History-driven gateways

In this section, we illustrate how the proposed approach can be used for expressing data and resource synchronization constraints. Moreover, we introduce a *decision* gateway, based on timelines, for specifying the decision rule about the branch to execute.

In Fig. 8, we report an example of a healthcare process for taking care of severely injured patients. The process of Fig. 8 involves three actors: *paramedics*, *nurses*, and *operating room staff*. Each actor is represented as a swimlane within the pool. *Paramedics* reach the patient and provide transport for her. *Nurses* take care of the patient when she arrives at the hospital, and *operating room staff* (i.e., surgeon and anesthetist), alerted in critical situations, provide emergency surgery.

This simple example allows us to introduce the following constraints on data, resources and decisions that may be naturally captured by means of timelines:

- *Enforce parallelization:* this constraint allows us to enforce the simultaneosly execution of two activities beloging to a parallel block. As an example, for specifying that if block bl_{loc} is chosen, then the execution of its internal loop bl_{calm} must be performed for the whole duration of the surgery, we can use the rule: $x[bl_{calm} = \top] \to \exists y[t_{sur} = \top](y \subseteq x)$. This kind of constraint is symmetrical with respect to the Disjoint Parallel Task constraint described in Sec. 5.
- *Message passing:* BPMN elements like messages, with their possible different semantics, may be easily integrated in our formalism. In this paper, for the sake of space, we only sketch an idea of this kind of constraints, without giving a detailed description and analysis. As an example, during the transport of the patient, paramedics may alert the operation room staff in case the patient situation is getting worse. The aim of the notification alert is requiring the preparation of the operating room. This mechanism is managed by the event-based gateway eg in Fig. 8 in the following way: (notice that we consider the messages attached to the gateway as part of it):

$$x[eg = \top] \to \begin{array}{l} \exists z[se_p = \top]\exists \overline{z}[sm_{op} = \bot]\exists y[t_{care} = \top]\exists \hat{y}[t_{care} = \bot]\exists \check{y}[t_{care} = \bot]\exists \overline{y}[bl_{room} = \bot] \\ \left(z \cap_{BMO} x \wedge \overline{z} \leq^{s,s}_{[0,+\infty)} x \wedge z \leq^{e,e}_{[0,+\infty)} \overline{z} \wedge \hat{y}\langle M\rangle y \wedge y\langle M\rangle \check{y} \wedge \hat{y} \cap_{BMO} x \wedge x \cap_{BMO} \check{y} \wedge y \subseteq x \wedge x \subseteq \overline{y} \right) \vee \\ \exists z[sm_{op} = \top]\exists \overline{z}[se_p = \top]\exists y[bl_{room} = \top]\exists \hat{y}[bl_{room} = \bot]\exists \check{y}[bl_{room} = \bot]\exists \overline{y}[t_{care} = \bot] \\ \left(z \subseteq x \wedge \overline{z} \leq^{s,s}_{[0,+\infty)} x \wedge z \leq^{e,e}_{[0,+\infty)} \overline{z} \wedge \hat{y}\langle M\rangle y \wedge y\langle M\rangle \check{y} \wedge \hat{y} \cap_{BMO} x \wedge x \cap_{BMO} \check{y} \wedge y \subseteq x \wedge x \subseteq \overline{y} \right) \end{array}$$

  This proposal is similar to an exclusive gateway with the addition, by means of the $z/\overline{z}$ variables, of a constraint regarding the preemptiveness of messages determining which block will be executed. Managing end events and intermediate message events is slightly different, since the former can be seen as the end of the related block. For example, in Fig. 8, *patient arrives* is the end of the block se_p. The duration of sm_{op} must be constrained to 1 unit, in order to make it istantaneous by means of rule $x[sm_{op} = \top] \to x \leq^{s,e}_{[1,1]} x$. Finally, when an intermediate message does not appear as a successor of an event-based gateway, its semantics must be explicitly encoded. This is the case of m_{pa} in Fig. 8 which is mapped to rule $x[m_{pa} = \top] \to \exists y[se_p = \top](x \leq^{e,e}_{[0,0]} y)$.
- *Resources and roles management:* a very important aspect to consider in managing Business Processes is related to the definition of roles

that are involved in the process execution. BPMN provides swimlanes (within pools) for representing roles (within organizations).

In the process of Fig. 8 tasks must be performed by the related roles (represented by means of swimlanes), this means that a paramedic cannot perform the surgery, and an anesthetist cannot drive the ambulance. However both a paramedic and an anesthetist may perform task t_{assess}. To force such constraint the rule $x[t_{assess} = \top] \rightarrow \exists y[Paramedic = \top](x = y) \vee \exists y[Anesthetist = \top](x = y)$ can be specified. In this case paramedics and anesthetists represent sets of timelines, one for each resource available in the considered instance. Each of such timelines represents how the specific resource is allocated to each task. The mutual exclusion in the use of resources is guaranteed by the non-overlapping nature of the intervals on the same timeline. The described notation allows us to abstract the number of available resources, since corresponding numbers are inserted at verification time. For example, by instantiating the above rule by using 2 paramedics and 3 anesthetists, we obtain $x[t_{assess} = \top] \rightarrow \exists y[paramedic_1 = \top](x = y) \vee \exists y[paramedic_2 = \top](x = y) \vee \exists y[anesthetist_1 = \top](x = y) \vee \exists y[anesthetist_2 = \top](x = y) \vee \exists y[anesthetist_3 = \top](x = y)$. This allows us to verify quantitative properties related to durations even in presence of multiple instances of the same process, that access the same resources [11].

- *Data driven decisions:* the described timeline-based approach is able to provide a preliminary integration of processes and data. Other proposals presented in literature [9, 13] are more focused on integrating existing formalisms (e.g., Entity-Relation data model), for representing data in BPMN process models.

  The timeline-based approach should not be considered as an alternative for such approaches, but as an annotation working well along with them, by helping in clarifying and verifying properties of data at the time execution of processes.

  As an example, let us suppose that the decision about which branch of $eg_{stable?}$ has to be chosen, is determined by both patient blood pressure (pBP), and patient body temperature (pBT). Let us assume that pBP and pBT are represented by means of the different timelines $(\mathbf{V}_{pBP}, \mathbf{\Delta}_{pBP}, \mathbf{D}_{pBP})$ and $(\mathbf{V}_{pBT}, \mathbf{\Delta}_{pBT}, \mathbf{D}_{pBT})$, respectively. More precisely, $\mathbf{V}_{pBP} = \{70, \ldots, 190, \bot\}$, i.e., on the timeline pBP all the possible ranges of blood pressures plus a disabled value when the pressure is not measured, are represented. The same holds for the body temperature, i.e., $\mathbf{V}_{pBT} = \{29, \ldots 41, \bot\}$. In this case, data may be available only when task t_{assess} is performed, thus the rules $x[t_{assess} = \bot] \rightarrow \exists y[pBP = \bot](y = x)$, $x[t_{assess} = \top] \rightarrow \bigvee_{v \in \{70,\ldots,190\}} \exists y[pBP = v](x = y)$ model this constraint. Similar rules

can be specified for constraining pBT.
Finally, for constraining the gateway sm_{op} to be executed whenever the values of BT and BP exceed certain thresholds, this set of rules can be specified:

$$
\begin{array}{l}
y[BP = v] \rightarrow \exists x[eg_{stable?} = \top]\exists z[sm_{op} = \top](y \subseteq x \wedge z \subseteq x) \vee \exists x[eg_{stable?} = \bot](y \subseteq x), \text{ with } v > 150 \\
y[BT = v] \rightarrow \exists x[eg_{stable?} = \top]\exists z[sm_{op} = \top](y \subseteq x \wedge z \subseteq x) \vee \exists x[eg_{stable?} = \bot](y \subseteq x), \text{ with } v > 39
\end{array}.
$$

This approach favours the compositionality in the addition of data to process. The described example is able to represent the way in which, by means of timeline-based annotation, it is possible to enrich processes with constraints on temporal aspects, roles and data. Process model are equipped both with the constraints on their execution and how they affect the process in both decisions and durations, without burden the process model.

- *Special behaviors for gateways:* in this work we show how to express BPMN diagram semantics and complex temporal constraints that would involve, if integrated directly in the diagram, complex patterns of throw/catch events as well as event-based gateways. We intentionally did not extend BPMN with some new element in order to stay within the boundaries of BPMN semantics. However, it is possible to use TS-RULES for extending the standard BPMN notation, by expressing the behavior of complex new elements in a straightforward way. As an example, let us consider gateway fg in Fig. 8. If it is the case that an instance of the process reaches fg, we expect that the patient has to be sedated, either totally or locally, while surgery has to be performed. Thus, in this case, we expect that branch t_{sur} is anyway executing, while choosing exactly one between the branches t_{tot} and bl_{loc}. Moreover, the choice between t_{tot} and bl_{loc} will be dictated by recent results of measurements related to the patient condition. For example, in case that $pBP > 150$ at some time point between 3 hours and the beginning of the surgery, a partial sedation has to be administered, otherwise it is possible to administer the total one. Rules for specifying these expectations are the following:

$$
\begin{array}{l}
x[fg = \top] \rightarrow \begin{array}{c}\exists y[t_{sur} = \top]\exists z[t_{tot} = \top]\exists w[bl_{loc} = \bot](y \subseteq x \wedge z \subseteq x \wedge x \subseteq w) \vee \\ \exists y[t_{sur} = \top]\exists z[bl_{loc} = \top]\exists w[t_{tot} = \bot](y \subseteq x \wedge z \subseteq x \wedge x \subseteq w)\end{array}, \\
x[pBP = v] \rightarrow \begin{array}{c}\exists y[fg = \top]\exists z[bl_{loc} = \top](x \leq^{e,s}_{[0,3\ hours]} y \wedge z \subseteq y) \vee \\ \exists y[fg = \bot](x \leq^{e,e}_{[3\ hours,+\infty)} y \wedge y \leq^{s,s}_{[0,+\infty)} x)\end{array}, \text{ with } v > 150
\end{array}.
$$

Summing up, by means of timelines we are able to introduce a BPMN element that behaves like a *conditional parallel* gateway, that is, a parallel gateway which runs all and only the branches that satisfy a certain condition at the precise moment of its execution.

7 Conclusion

In this paper we dealt with issues related to the specification of different kinds of constraints on process models represented by means of BPMN diagrams. We provided a timelines-based approach for expressing admissible executions of a process. Timelines allow us to give complex constraints possibly related to time, data, and resources, by annotating the BPMN process diagram, without overburden the process diagram itself. Some of the advantages of our proposal are (i) providing a means for specifying complex constraints without extending BMPN; (ii) applying the existing tools for timeline-based planning [2, 5, 6] for verifying *qualitative properties* at design time; (iii) supporting resources optimization in the style of [11], and (iv) checking *quantitative properties* such as interplay between the number of mutually exclusive resources and the number of process instances that may be completed in a given amount of time. For future work, we plan to apply synchronization rules for querying running processes and monitoring business process activities (Business Activity Monitoring (BAM)[3]).

References

[1] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, November 1983.

[2] Javier Barreiro, Matthew Boyce, Minh Do, Jeremy Frank, Michael Iatauro, Tatiana Kichkaylo, Paul Morris, James Ong, Emilio Remolina, Tristan Smith, and David Smith. EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization. In *ICKEPS 4*.

[3] Catriel Beeri, Anat Eyal, Tova Milo, and Alon Pilberg. Query-based monitoring of bpel business processes. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1122–1124. ACM, 2007.

[4] Yuval Shahar Carlo Combi, Elpida Keravnou-Papailiou. *Temporal Information Systems in Medicine.* Springer Science & Business Media, 2010.

[5] Amedeo Cesta, Gabriella Cortellessa, Simone Fratini, and Angelo Oddi. Developing an end-to-end planning application from a timeline representation framework. 2009.

[6] S. Chien, D. Tran, G. Rabideau, S.R. Schaffer, D. Mandl, and S. Frye. Timeline-based space operations scheduling with external constraints. In *Proc. of the 20^{th} International Conference on Automated Planning and Scheduling*, pages 34–41, 2010.

[7] Marta Cialdea Mayer, Andrea Orlandini, and Alessandro Umbrico. Planning and execution with flexible timelines: a formal account. *Acta Informatica*, 53(6):649–680, Oct 2016.

[8] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *International Conference on Computer Aided Verification*, pages 359–364. Springer, 2002.

[9] Carlo Combi, Barbara Oliboni, Mathias Weske, and Francesca Zerbato. Conceptual modeling of processes and data: Connecting different perspectives. In *ER 2018*, volume 11157 of *LNCS*, pages 236–250, 2018.

[10] Carlo Combi, Pietro Sala, and Francesca Zerbato. Driving time-dependent paths in clinical BPMN processes. In *Proceedings of SAC 2017*, pages 743–750, 2017.

[11] Carlo Combi, Pietro Sala, and Francesca Zerbato. A logical formalization of time-critical processes with resources. In *BPM Forum 2018*, volume 329 of *LNBIP*, pages 20–36. Springer, 2018.

[12] Giuseppe De Giacomo, Marlon Dumas, Fabrizio Maria Maggi, and Marco Montali. Declarative process modeling in bpmn. In *CAISE*, pages 84–100. Springer, 2015.

[13] Riccardo De Masellis, Chiara Di Francescomarino, Chiara Ghidini, Marco Montali, and Sergio Tessaris. Add data into business process verification: Bridging the gap between theory and practice. In *Proc. of 31^{st} AAAI Conference on Artificial Intelligence*, pages 1091–1099. AAAI Press, 2017.

[14] Marlon Dumas, Luciano García-Bañuelos, and Artem Polyvyanyy. Unraveling unstructured process models. In Jan Mendling, Matthias Weidlich, and Mathias Weske, editors, *Business Process Modeling Notation*, pages 1–7, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[15] Jan-Philipp Friedenstab, Christian Janiesch, Martin Matzner, and Oliver Muller. Extending bpmn for business activity monitoring. In *2012 45th Hawaii International Conference on System Sciences*, pages 4158–4167. IEEE, 2012.

[16] Gerard J Holzmann. *The SPIN model checker: Primer and reference manual*, volume 1003. Addison-Wesley Reading, 2004.

[17] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems.* Cambridge University Press, 2004.

[18] Bartek Kiepuszewski, Arthur H. M. ter Hofstede, and Christoph Bussler. On structured workflow modelling. In Benkt Wangler and Lars Bergman, editors, *Advanced Information Systems Engineering, 12th International Conference CAiSE 2000, Stockholm, Sweden, June 5-9, 2000, Proceedings*, volume 1789 of *Lecture Notes in Computer Science*, pages 431–445. Springer, 2000.

[19] Andreas Lanz, Roberto Posenato, Carlo Combi, and Manfred Reichert. Controlling time-awareness in modularized processes. *Enterprise, Business-Process and Information Systems Modeling*, pages 157–172, 2016.

[20] Andreas Lanz, Manfred Reichert, and Barbara Weber. Process time patterns: A formal foundation. *Information Systems*, 57:38 – 68, 2016.

[21] Fabrizio Maria Maggi, Marco Montali, Michael Westergaard, and Wil M. P. van der Aalst. Monitoring business constraints with linear temporal logic: An approach based on colored automata. In *BPM 2011.*, volume 6896 of *LNCS*, pages 132–147. Springer, 2011.

[22] Leonard A Mermel, Barry M Farr, Robert J Sherertz, Issam I Raad, Naomi O'grady, JoAnn S Harris, and Donald E Craven. Guidelines for the management of intravascular catheter-related infections. *Infection Control & Hospital Epidemiology*, 22(4):222–242, 2001.

[23] Steven Mertens, Frederik Gailly, and Geert Poels. Enhancing declarative process models with DMN decision logic. In Khaled Gaaloul, Rainer Schmidt, Selmin Nurcan, Sérgio Guerreiro, and Qin Ma, editors, *Enterprise, Business-Process and Information Systems Modeling - 16th International Conference, BPMDS 2015, 20th International Conference, EMMSAD 2015, Held at CAiSE 2015, Stockholm, Sweden, June 8-9, 2015, Proceedings*, volume 214 of *Lecture Notes in Business Information Processing*, pages 151–165. Springer, 2015.

[24] Angelo Montanari and Pietro Sala. Interval-based synthesis. In Adriano Peron and Carla Piazza, editors, *Proceedings Fifth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2014, Verona, Italy, September 10-12, 2014.*, volume 161 of *EPTCS*, pages 102–115, 2014.

[25] Nicola Muscettola. Hsts: Integrating planning and scheduling. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1993.

[26] OMG. Business Process Model and Notation (BPMN), Version 2.0, January 2011.

[27] Roberto Posenato, Francesca Zerbato, and Carlo Combi. Managing decision tasks and events in time-aware business process models. In *BPM 2018.*, volume 11080 of *LNCS*, pages 102–118. Springer, 2018.

[28] Wolfgang Thomas. On the synthesis of strategies in infinite games. In *STACS*, pages 1–13, 1995.

[29] Petia Wohed, Wil MP van der Aalst, Marlon Dumas, Arthur HM ter Hofstede, and Nick Russell. On the suitability of bpmn for business process modelling. In *International conference on business process management*, pages 161–176. Springer, 2006.

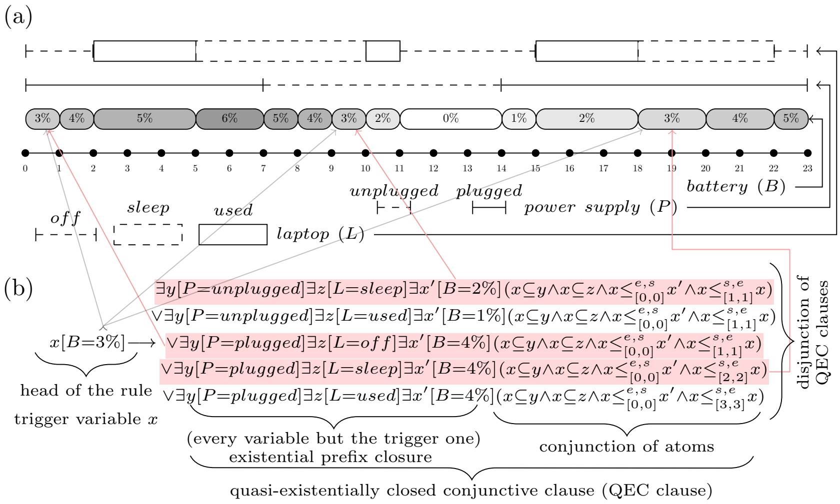

Figure 9: An example of formalization for laptop energy consumption using the timelines-based formalism.

A Timelines by Example

In this section we introduce the basic concepts of timelines and of timelines-based planning [25] by means of an example. Since formal description of timeline-based formalisms may be found in literaure (as an example see [7]) and, briefly, in Sec. 4.

Let us suppose that we want to model the energy consumption estimation for a battery of a laptop computer. In such a scenario we have the following components: (i) the *battery* B of the laptop having a level of charge going from 0% to 100%; (ii) the *power supply* P of the laptop that may be *unplugged* or *plugged* to the laptop, in this latter case we always assume that the laptop is charging the battery; (iii) the *laptop* has three possible states, it may be *used* by someone, may be put in *sleep* for energy saving, or may be turned off.

For the sake of simplicity and compactness, we measure time into abstract time units (e.g., you may see a single time unit as minutes), then time domain is isomorphic to $\mathbb{N}$. In this context the constraints on the battery level can be: (i) if the power supply is *plugged* and the laptop is turned off then the battery gets an increment (clearly if it is not already at level 100%) of its level by 1% every minute, if the laptop is in *sleep* state it gets an increment of its level by 1% every 2 minutes, and every 3 minutes if the laptop is *used*; (ii) if the power supply is *unplugged* and the laptop is turned off the level of the battery stays stationary, while if it is sleep it gets a decrement of 1% every minutes, and a decrement of 2% every minute if laptop is *used*; (iii) a laptop with battery at 0% cannot be neither *used* nor in *sleep* (i.e., it is turned off).

In this case, we do not consider fractions in the time of charge. For instance, if the computer is plugged and used for 1 or 2 minutes before we put it on *stand_by* we simply did not consider the increment. In such a way we have a conservative estimation of the battery level, that is, due to residuals the battery may be really charged more than how it is modelled but never less than it.

If we focus on the single components, namely battery, power_supply, and laptop, we associate to every of such components a *state variable* which is a triple $sv = (\mathbf{V}_{sv}, \mathbf{\Delta}_{sv}, \mathbf{D}_{sv})$, where $\mathbf{V}_{sv}$ is the finite domain of the state variable sv, $\mathbf{\Delta}_{sv} : \mathbf{V}_{sv} \to 2^{\mathbf{V}_{sv}}$ is a transition function, and $\mathbf{D}_{sv} : \mathbf{V}_{sv} \to \mathbb{N} \times \mathbb{N} \cup \{+\infty\}$ is a duration function.

In our example we associate each state variable with every component, thus we have state variables B, P and L. Then, we have $\mathbf{V}_B = \{1\%, \ldots, 100\%\}$, $\mathbf{V}_P = \{plugged, unplugged\}$, and $\mathbf{V}_L = \{off, sleep, used\}$. Transition functions for the state variables determine the set of values that are allowed after a certain value has been taken. In our example we have that the battery may stay stationary, decrease of 1/2 units or increase by one unit, thus its transition function is $\mathbf{\Delta}_B(v) = \begin{cases} \{v-2, v-1, v, v+1\} & 1\% < v < 100\% \\ \{v-1, v, v+1\} & v = 1\% \\ \{v, v+1\} & v = 0\% \\ \{v-1, v\} & v = 100\% \end{cases}$, while the transitions fuctions for state variables P and L are $\mathbf{\Delta}_P(plugged) = \{unplugged\}$, $\mathbf{\Delta}_P(unplugged) = \{plugged\}$, $\mathbf{\Delta}_L(used) = \{sleep, off\}$, $\mathbf{\Delta}_L(sleep) = \{used, off\}$, and $\mathbf{\Delta}_L(off) = \{used, sleep\}$.

The duration function associates to each value in the domain of the state variable a subinterval $[l, u]$ of $[1, +\infty)$ which represents the bounds of duration of the given value. In our example we do not have such constraints, thus $\mathbf{D}_*(v) = [1, +\infty)$ for every $* \in \{B, P, L\}$ and every $v \in \mathbf{V}_*$. For instance, for expressing the fact that *laptop* may not be used consecutively more than 5 hours straight in order to preserve monitor life, we should have put $\mathbf{D}_L(used) = [1, 300]$.

A timeline for a state variable sv is a sequence of *tokens*. A token for a state variable sv is a pair (v, d) where $v \in \mathbf{V}_{sv}$ and $d \in \mathbf{D}_{sv}(v)$. In Fig. 9.(a) an example of timelines for the state variables of our example is reported. For instance, for B the timeline is $\langle(3\%, 1), (4\%, 1), (5\%, 3), (6\%, 2), (5\%, 1), (4\%, 1), (3\%, 1), (2\%, 1), (0\%, 3), (1\%, 1), (2\%, 3), (3\%, 2)(4\%, 2), (5\%, 1)\rangle$ while for P the timeline is $\langle(plugged, 7), (unplugged, 7), (plugged, 9)\rangle$.

Synchronization between timelines is provided by a set of *temporal synchronization rules*. The structure of a temporal synchronization rule is shown in Fig. 9.(b). In particular, in Fig. 9.(b) the rule that constrains the case in which the battery has level 3% is described. Similar rules may be written using an analogous template. A temporal synchronization rule has the form of an implication. The tail of such implication is formed by a variable, called *trigger variable*, (x in Fig. 9.(b)) and an assignment to a state variable ($B = 3\%$ in Fig. 9.(b)). The variable ranges over the tokens in the timeline of its state variable and whenever the value associated to a token is the same to the one of the assignment the body of the rule is triggered. In other words, the variable in the tail of the implication is universally quantified. In Fig. 9.(a), we may observe that the rule in Fig. 9.(b) is triggered three times (i.e., there are three tokens on the B timeline with value 3%). The head of the rule is a *disjunctions* of possible scenarios represented by a set of existentially closed *conjunctive clauses* of atoms in which the only free variable may be the triggered one. An *atom* is a binary relation between the intervals represented by the two tokens associated to two variables. For instance, the first clause in Fig. 9.(b) says that, if there exists a token in the timeline of *power* with value *unplugged* (variable y), and a token on the timeline of *laptop* with value *sleep* (variable z), both containing the token associated to x (atoms $x \subseteq y$ and $x \subseteq z$), then the token associated to x has length 1 (atom $x \leq^{s,e}_{[1,1]} x$) and there exists a token for the *battery* (variable x') that follows the token for x (atom $x \leq^{[e,s]}_{[0,0]} x'$) with value 2%. This is exactly the situation that occurs at the second occurrence of 3% on the timeline for B in Fig. 9.(a) and thus the rule is satisfied for that token. It is easy to see, by means of different disjuncts, that all the occurrences of 3% are fulfilled by the timelines in Fig. 9.(b), and thus the rule is satisfied by the timelines. Given a set of temporal synchronization rules, a *timeline-based planning problem* consists of determining whether or not there exists a set of timelines that satisfies all of them.

B Structured BPMN Diagrams semantics via Timelines

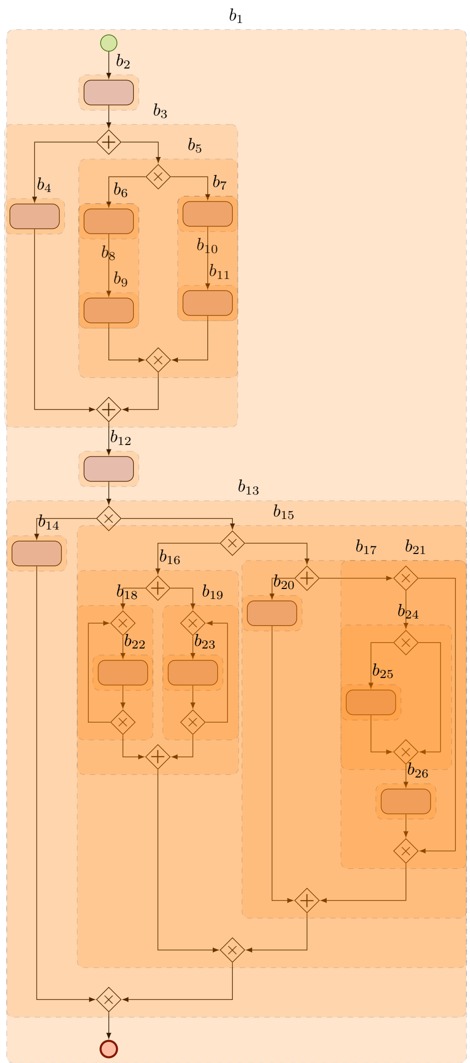

Figure 10: The set $Blocks_{\mathbf{D}}$ relative to the structured diagram represented of Fig. 2.

In this section, first we provide a timeline-based semantics for structured BPMN Diagrams, then we provide how the annotations proposed in Section 5 are mapped at this level of detail. For space reason we will consider only significant set of BPMN core elements (the translation of the missing BPMN elements will addressed in an extended version of the paper).

As we told before we restrict ourselves to structured diagrams [14]. Structured diagrams $\mathbf{D}$ may be partitioned into a set of Single-Entry-Single-Exit (SESE) blocks $Blocks_{\mathbf{D}} = \{b_1, \ldots, b_n\}$ such that its blocks are pairwise contained or disjoint. Fig. 10 it is represented the set $Blocks_{\mathbf{D}} = \{b_1, \ldots, b_{26}\}$ relative to the structured diagram $\mathbf{D}$ of Fig. 2.

Given a SESE block b we define its *entry point* (resp., *exit point*), denoted by $\overrightarrow{b}$ (resp. $\underrightarrow{b}$), as the BPMN element receiving its incoming (resp. outgoing) flow edge.

Now we provide a translation of a structured BPMN diagram $\mathbf{D}$ into a set of TS-RULES via a function $\mathcal{R}ules_{\mathbf{D}} : \mathbf{D} \rightarrow 2^{TS-\text{RULES}}$. All the TS-RULES in the image of $\mathcal{R}ules_{\mathbf{D}}$ will speak about a subset of state variables $b_1, \ldots, b_n$ (i.e., one state variable for each block). For each $1 \leq i \leq n$ we have $\mathbf{V}_{b_i} = \{\top, \bot\}$, where $\top$ (resp., $\bot$) labels token in which b_i is (resp., is not) currently being executed, $\boldsymbol{\Delta}_x(\top) = \{\bot\}$, $\boldsymbol{\Delta}_x(\top) = \{\bot\}$, and $\mathbf{D}_x(\top) = \mathbf{D}_x(\bot) = (1, +\infty)$. Informally speaking, each state variable of type b_i is an $\top/\bot$ switch where $\top$ (resp. $\bot$) tokens may take any duration different from 0.

Let us now introduce a classification on the possible types of SESE blocks together with the set TS-RULES intoduced by mapping $\mathcal{R}ules_{\mathbf{D}}$:

TASK blocks sorrounding single tasks like the blocks $b_6, b_7, b_9, b_{11}, b_{12}, b_{14}, b_{20}, b_{22}, b_{23}, b_{25}$, and b_{26} in Fig. 10. They are the basic starting blocks and their presence do not introduce any TS-RULE in $\mathcal{R}ules_{\mathbf{D}}$ (i.e., $\mathcal{R}ules_{\mathbf{D}}(b) = \emptyset$ for every $b \in Blocks_{\mathbf{D}}$ which is of type **TASK**);

FLOW these blocks are blocks which enclose maximal paths of blocks $b_{i_1}, \ldots, b_{i_m}$ with $m > 1$, such that there exists a flow edge between $\underrightarrow{b_{i_j}}$ and $\overrightarrow{b_{i_{j+1}}}$ for every $1 \leq j < m$. In Fig. 10 we have that initial block b_1 as well as blocks b_8 and b_{10} are of type **FLOW**. If b is of type **FLOW** we denote with $Path_b$ the sequence of blocks it encloses, i.e., $Path_b = b_{i_1} \ldots b_{i_m}$. In Fig. 10 we have $Path_{b_1} = b_2 b_3 b_{12} b_{13}$, $Path_{b_8} = b_6 b_9$ and $Path_{b_{10}} = b_7 b_{11}$. For every **FLOW** $b \in Blocks_{\mathbf{D}}$ let $path(b) = b_{i_1}, \ldots, b_{i_m}$ we define $\mathcal{R}ules_{\mathbf{D}}(b)$ as the following set of TS-RULES:

$$a)\ \{x[b = \top] \rightarrow \exists y[b_{i_j} = \top]\exists \hat{y}[b_{i_j} = \bot]\exists \check{y}[b_{i_j} = \bot]\, (\hat{y}\langle M\rangle y \wedge y\langle M\rangle \check{y} \wedge \hat{y}\langle O\rangle x \wedge y\langle D\rangle x \wedge x\langle O\rangle \check{y}) : 1 < j < m\}$$

$$\cup$$

$$b)\ \left\{x[b = \top] \rightarrow \begin{array}{c} \exists y[b_{i_1} = \top]\exists \hat{y}[b_{i_1} = \bot]\exists \check{y}[b_{i_1} = \bot]\, (\hat{y}\langle M\rangle y \wedge y\langle M\rangle \check{y} \wedge \hat{y}\langle O\rangle x \wedge y\langle D\rangle x \wedge x\langle O\rangle \check{y}) \vee \\ \exists y[b_{i_1} = \top]\exists \check{y}[b_{i_1} = \bot](y\langle M\rangle \check{y} \wedge y\langle B\rangle x \wedge x\langle O\rangle \check{y}) \end{array}\right\}$$

$$\cup$$

$$c)\ \left\{x[b = \top] \rightarrow \begin{array}{c} \exists y[b_{i_m} = \top]\exists \hat{y}[b_{i_m} = \bot]\exists \check{y}[b_{i_m} = \bot]\, (\hat{y}\langle M\rangle y \wedge y\langle M\rangle \check{y} \wedge \hat{y}\langle O\rangle x \wedge y\langle D\rangle x \wedge x\langle O\rangle \check{y}) \vee \\ \exists y[b_{i_m} = \top]\exists \hat{y}[b_{i_m} = \bot](\hat{y}\langle M\rangle y \wedge y\langle E\rangle x \wedge \check{y}\langle O\rangle x) \end{array}\right\}$$

$$\cup$$

$$d)\ \left\{x[b = \top] \rightarrow \exists y[b_{i_j} = \top]\exists y'[b_{i_{j+1}} = \top](y\langle D\rangle x \wedge y'\langle D\rangle x \wedge y \leq^{e,s}_{[0,+\infty)} y') : 1 \leq j < m\right\}$$

EXCLUSIVE these blocks are blocks b whose entry point (resp., exit point) $\overrightarrow{b}$ (resp., $\underrightarrow{b}$) is an exclusive split gateway (resp. exclusive join gateway). For instance in Fig. 10 we have that blocks $b_5, b_{13}, b_{15}, b_{21}$, and b_{24} are of type **EXCLUSIVE**. For every **EXCLUSIVE** b let b_1 and b_2 be the two maximal blocks contained in b and connected to $\overrightarrow{b}$, we define $\mathcal{R}ules_{\mathbf{D}}(b)$ as the following (singleton) set of TS-RULES:

$$\left\{x[b = \top] \rightarrow \begin{array}{c} \exists y[b_1 = \top]\exists \hat{y}[b_1 = \bot]\exists \check{y}[b_1 = \bot]\exists \overline{y}[b_2 = \bot](\hat{y}\langle M\rangle y \wedge y\langle M\rangle \check{y} \wedge \hat{y} \cap_{BMO} x \wedge x \cap_{BMO} \check{y} \wedge y \subseteq x \wedge x \subseteq \overline{y}) \\ \vee \\ \exists y[b_2 = \top]\exists \hat{y}[b_2 = \bot]\exists \check{y}[b_2 = \bot]\exists \overline{y}[b_1 = \bot](\hat{y}\langle M\rangle y \wedge y\langle M\rangle \check{y} \wedge \hat{y} \cap_{BMO} x \wedge x \cap_{BMO} \check{y} \wedge y \subseteq x \wedge x \subseteq \overline{y}) \end{array}\right\}$$

LOOP these blocks are blocks b whose entry point (resp., exit point) $\overrightarrow{b}$ (resp., $\underrightarrow{b}$) is an exclusive join gateway (resp. exclusive split gateway). For instance in Fig. 10 we have that blocks b_{22} and b_{23} are of type **LOOP**. For every **LOOP** let b_1 be the maximal block contained in b and connected to $\underrightarrow{b}$ we define $\mathcal{R}ules_{\mathbf{D}}(b)$ as the following (singleton) set of TS-RULES:

$$\left\{x[b = \top] \rightarrow \exists \hat{y}[b_1 = \bot]\exists y[b_1 = \top]\exists y'[b_1 = \top]\exists \check{y}[b_1 = \bot] \left(\begin{array}{c} \hat{y} \leq^{s,s}_{[0,+\infty)} x \wedge x \leq^{s,e}_{[0,+\infty)} \hat{y} \wedge \hat{y}\langle M\rangle y \wedge y \subseteq x \wedge \\ \check{y} \leq^{s,e}_{[0,+\infty)} x \wedge x \leq^{e,e}_{[0,+\infty)} \check{y} \wedge y'\langle M\rangle \check{y} \wedge y' \subseteq x \end{array} \right)\right\}$$

PARALLEL these blocks are blocks b whose entry point (resp., exit point) $\overrightarrow{b}$ (resp., $\underrightarrow{b}$) is an parallel split gateway (resp., parallel join gateway). For instance in Fig. 10 we have that blocks $b_e, b_{13}, b_{15}, b_{21}$, and b_{24} are of type **PARALLEL**. For every **PARALLEL** b let b_1 and b_2 be the two maximal blocks contained in b and connected to $\overrightarrow{b}$, we define $\mathcal{R}ules_{\mathbf{D}}(b)$ as the following (singleton) set of TS-RULES:

$$\left\{ x[b=\top] \rightarrow \begin{array}{c} \exists y[b_1=\top]\exists\hat{y}[b_1=\bot]\exists\check{y}[b_1=\bot]\exists y'[b_2=\top]\exists\hat{y}'[b_2=\bot]\exists\check{y}'[b_2=\bot] \\ \left(\begin{array}{c} \hat{y}\langle M\rangle y \wedge y\langle M\rangle \check{y} \wedge \hat{y} \leq^{s,s}_{[0,+\infty)} x \wedge x \leq^{s,e}_{[0,+\infty)} \hat{y} \wedge y \subseteq x \wedge \check{y} \leq^{s,e}_{[0,+\infty)} x \wedge x \leq^{e,e}_{[0,+\infty)} \check{y} \wedge \\ \hat{y}'\langle M\rangle y' \wedge y'\langle M\rangle \check{y}' \wedge \hat{y}' \leq^{s,s}_{[0,+\infty)} x \wedge x \leq^{s,e}_{[0,+\infty)} \hat{y}' \wedge y' \subseteq x \wedge \check{y}' \leq^{s,e}_{[0,+\infty)} x \wedge x \leq^{e,e}_{[0,+\infty)} \check{y}' \end{array} \right) \end{array} \right\}$$