High-Level Requirements-Driven Business Process Compliance

Juanita Caballero-Villalobos[1][0000−0002−4915−0961], Andrea Burattin[1][0000−0002−0837−0183], and Hugo A. López[1][0000−0001−5162−7936]

Technical University of Denmark, Kongens Lyngby 2800, Denmark
{jcavi, andbur, hulo}@dtu.dk

Abstract. Process compliance refers to the alignment between business processes and regulatory requirements. Compliance is difficult because it needs to be able to express the intent and the possible interpretations of laws into formal models, align models with traces in a process, and inspect whether these traces are generating violations. This paper focuses on a largely unexplored area within BPM: the compliance of high-level and non-functional requirements. While compliance checking has been studied through conformance checking techniques, most regulatory requirements are defined in subjective and high-level terms, limiting the application of rule-checking and alignments to specific cases. In contrast, we propose the application of requirement engineering methods for business process compliance. In particular, we raise the level of abstraction from the compliance of specific patterns to the satisfaction of high-level goals and subjective qualities. We propose a framework that connects process models with goal models, rendering explicit alternatives for the satisfaction of vague goals and subjective qualities. Compliance checking is reduced to a reachability of a state where subjective qualities are satisfied. This approach is exhibited in a data protection scenario, and we provide a prototypical implementation of the compliance checking tool.

Keywords: Compliance Checking · Business Process Compliance · Goal Modeling · Requirements Engineering.

1 Introduction

Business processes are considered the heart of organizations. Through processes, companies achieve objectives, coordinate and optimize resources, and comply with regulatory requirements. Legal compliance became a substantial task for all business organizations. Yet, the majority of organizations treat law as an exogenous force and compliance is mapped and measured rather than explained [23]. The management of business processes requires traceability between high-level requirements (for instance, laws) to traces in an information system. Regulatory compliance is one of the major drivers behind the adoption of process mining in the industry [14], yet there is still a large gap between regulations and the artifacts used in PM. In particular, there is a non-trivial interpretative factor: a legal

paragraph may have multiple interpretations *by-design* [10], and its disambiguation may therefore require human support. This contrasts with the intention of formalizing policies with a single, mathematical, and unequivocal semantics.

Modeling high-level business requirements is challenging, as their disambiguation must be resolved by experts in both the process and legal domains [20], and if those experts do not remain involved through later compliance assessments, the rationale behind each interpretation is lost. Multiple techniques have been applied to resolve legal compliance, including combinations of imperative models and modal logics [13], conformance checking [5], and declarative process models [21] (see [22] for a recent review). However, these works consider a low-level view of compliance, where goals in an organization can be directly mapped to activities in a process. This limits the applicability to regulatory compliance, where most requirements come in the form of high-level and subjective descriptions. For instance, consider the following excerpt: *"the company needs to send a **timely** delivery confirmation"*. Existing approaches may verify that a confirmation was sent; however, they will not be able to deal with the ambiguities generated by not being able to define how "timely" delivery confirmation was.

This work explores how goal-modelling frameworks may help in the definition of compliance checking techniques to align high-level requirements and business processes. Goal Models [29] are a well-established set of techniques used in requirements engineering to capture non-functional requirements (see [7] for a recent guide). Thanks to their graphical representation, it allows the communication of multiple stakeholders about functional and non-functional requirements. While the relation of Goal Models and Business Processes has been explored before (i.e., [12, 19, 24, 26, 28]), its use has been limited to top-down, simulation, and monitoring approaches, not considering qualities and non-functional requirements. Moreover, a typical pitfall of these works is the tight links between requirements and processes, when in reality, they evolve in different lifecycles [3].

In particular, we propose a compliance framework where 1) functional and non-functional requirements are modelled as IStar models [29], 2) any process modelling notation with an LTS semantics captures business processes, and 3) a mapping between requirements and tasks in a goal model and activities in a process model is maintained. This framework allows us to decouple the goals and processes, maintaining their independence, identify ways of satisfying high-level and non-functional goals, and reuse the goal models across multiple process notations. Moreover, we introduce a design-time compliance checking algorithm that evaluates the synchronized execution steps of process and goal models. As an example of the applicability of the framework, we use workflow nets as a process language, but the framework could be instantiated in imperative or declarative languages with an operational semantics, for instance, BPMN [9] and DCR graphs [16]. We illustrate the framework using a simple data protection guideline and provide a prototypical implementation of the framework.

Structure of this paper. Section 2 introduces our compliance framework; Section 3 presents the preliminaries; Section 4 provides a technique to check the

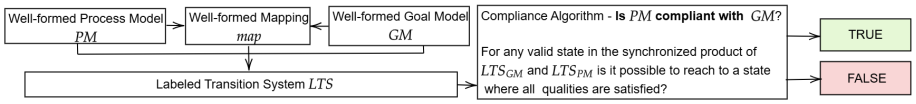


Fig. 1: Overview of the framework

compliance of high-level requirements; Section 5 details a prototypical implementation; Section 6 presents related work; and Section 7 concludes.

2 Approach

Figure 1 illustrates our compliance checking approach, which combines three well-formed inputs: the process model, the goal model, and the mapping. The process model encodes the *how*, while the goal model captures *what* and *why*, including the high-level business requirements (HL-BR). Both are translated into labeled transition systems (LTS) and synchronized via the mapping, yielding a composite system LTS_C. Compliance holds if, from every state in LTS_C, there exists a path to a state where all HL-BR are satisfied. Formal semantics and well-formedness conditions are detailed in the following section.

2.1 Running Example

To illustrate how our framework works, consider a fictitious scenario:

> **"(R) Every time a *company* releases a new feature, it must take *appropriate measures* to ensure its *data remains protected.*** The process begins with the IT department implementing a password policy by (a) *update the encryption standards* and (b) *revise access controls*. Then, the company deploys a system to monitor data security, choosing between (c) *deploy DataGrail or (d) OneTrust*. Once the monitoring system is in place, the security team (e) *conducts penetration tests* to detect potential vulnerabilities. Testing is repeated until no vulnerabilities are found. If there are anomalies, the system (f) *flags the suspicious activity*, and (g) *applies a vulnerability patch* to address the vulnerability. If no vulnerability is found, the testing is successful."

Determining the specific tasks to achieve R requires expert interpretation, and it may vary according to the understanding of "*appropriate measures*". Most business process compliance techniques [22] focus on verifying whether the tasks carried out align with the predefined ones (i.e., a, b, c, d, e, f, g). However, they do not distinguish among the conformant traces, which are more desirable to fulfill stakeholders' goals or differences in outcome quality [4]. For instance, assume two process executions both follow tasks (a) to (g), yet in one case, the company chooses *DataGrail* and in the other *OneTrust*. Although both traces are conformant, they may differ in terms of effectiveness or cost. Existing compliance techniques would treat them as equivalent, despite these distinctions.

3 Preliminaries

This section provides the background to understand and position the contributions of the framework. Section 3.1 provides a formalization of abstract models, capturing the commonalities between imperative process models and goal-oriented model languages. Section 3.2 and 3.3 instantiate the framework using i* for goal modeling and Workflow nets for process modeling.

3.1 Abstract Models

Any model artifact from a model-driven approach with trace-based semantics can serve as a basic model, provided it uses the same notation.

Definition 1 (Abstract Model Notation (adapted from [8])). *Let A be a fixed universe of model actions, where $a \in A$ is an action. An abstract model notation $\mathcal{MN} \triangleq \langle \mathcal{M}, alph, excluded, step \rangle$ comprises a set of models $\mathcal{M}$; a labeling function $alph : \mathcal{M} \to 2^A$; an exclusion function $excluded : \mathcal{M} \to 2^A$; and a transition predicate $\boldsymbol{step} \subseteq \mathcal{M} \times A \times \mathcal{M}$. Let M_1 and M_2 be two models in $\mathcal{M}$. We require that $\langle M_1, a, M_2 \rangle \in \boldsymbol{step}$ implies both $a \in alph(M_1)$ and $alph(M_1) = alph(M_2)$, and if also $\langle M_1, a, M_2' \rangle \in \boldsymbol{step}$ then $M_2 = M_2'$, then $\boldsymbol{step}$ is action-deterministic.*

Let $\mathcal{M}$ be a set of process models. Intuitively, *alph* bounds the actions a process may exhibit, and this bound must be preserved by ***step*** transitions. Similarly, *exclude* identifies actions excluded from a given process and may change over time. For non-monotonic languages, setting $excluded = \emptyset$ suffices.

We use Labeled Transition Systems (LTS) to capture configurations, actions, and transitions, where runs represent execution and traces their action sequence.

Definition 2 (Abstract Labeled Transition System). *Let $\mathcal{MN} = \langle \mathcal{M}, alph, excluded, step \rangle$ be an abstract model notation. $M \in \mathcal{M}$ is a model instance. An LTS of the model M, $LTS_M \triangleq \langle S_M, Act_M, \longrightarrow_M, s_0^M, F_M \rangle$ comprises a set of states S_M; a set of actions $Act_M = alph(M)$; a transition relation $\longrightarrow_M \subseteq S_M \times Act_M \times S_M$ is the transition relation, where $\langle M, a, M' \rangle \in \longrightarrow_M$ iff $\langle M, a, M' \rangle \in \boldsymbol{step}$; and initial state $s_0^M = M$; and a set of final states $F_M \subseteq S_M$.*

Definition 3 (Run and traces). *Let $LTS_M = \langle S_M, Act_M, \longrightarrow_M, s_0^M, F_M \rangle$ be the LTS of a model M and s_0 be its initial state. A* finite run *is a sequence $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} s_n$ with $s_i \xrightarrow{a_{i+1}} s_{i+1}$ for all $0 \leq i < n$. An* infinite run *continues indefinitely. A* trace σ *is the sequence $\langle a_1, a_2, \ldots \rangle$ of actions in the run.*

Let $LTS_M = \langle S_M, Act_M, \to_M, s_0^M, F_M \rangle$ be the LTS of a model M with $Act_M = Act_M \cup \{\epsilon\}$, where ϵ denotes no observable action. For $s, s' \in S_M$ and $a \in Act_M$, we write $s \xrightarrow{a}_M s'$ if $(s, a, s') \in \to_M$. The abstract reachability relation $s \xrightarrow{\sigma}{}^*_M s'$ holds for a finite sequence $\sigma = \langle a_1, \ldots, a_n \rangle$ such that $s = s_0 \xrightarrow{a_1}_M \cdots \xrightarrow{a_n}_M s_n = s'$. Reflexivity means $s \xrightarrow{\epsilon}_M s$, and transitivity means if $s \xrightarrow{\sigma_1}{}^*_M s'$ and $s' \xrightarrow{\sigma_2}{}^*_M s''$, then we write $s \xrightarrow{\sigma_1\sigma_2}{}^*_M s''$ to indicate that s'' is reachable from s via the execution of σ_1 and σ_2.

3.2 Goal Models

A goal model captures the *rationale* behind the execution of certain tasks, and *what* requirements a system should achieve, using goals, qualities, and tasks.

Definition 4 (Goal Model Elements). *Let the finite set of goal model elements be the tuple* $GE \triangleq \langle IE, L \rangle$*, where* $IE = T \cup G \cup Q$ *is a finite set of* intentional elements*, consisting of a set of tasks* T*, a set of goals* G*, and a set of qualities* Q*. The set* $L = R \cup C$ *denotes* intentional element relations *(or* links*), where* $R \subseteq (G \cup T) \times (G \cup T) \rightharpoonup \{and, or\}$ *is a set of refinement links, and* $C \subseteq (G \cup T) \times Q \rightharpoonup \{Make, Break\}$ *is a set of contribution links.*

We adopt naming conventions from [11] to ensure clarity and consistency in modelling: goals use passive syntax (i.e., *Data protected*), tasks use active forms (i.e., *Implement password policy*), and qualities add manner complements (i.e., *Data protected appropriately*). The model is further refined through links of the form $r(e_1, e_2)$, where element e_1 is refined by e_2. Figure 2, shows the refinement of the goal *Data Protected* (e_1) by tasks such as *Implement password policy* (e_2). Achieving e_1 contributes to fulfilling the HL-BR expressed as the quality *Appropriate measures to protect data*. Intentional elements (IE) are assigned to a truth state for reasoning over their satisfaction.

Definition 5 (Intentional Element Status). *Let* $GE = \langle IE, R \rangle$ *be the set of goal model elements, with* $IE = G \cup T \cup Q$ *(Definition 4). Let* $\Delta = \{\top, \bot, ?\}$ *be the set of truth-values "true," "false," and "unknown". Let* $d \in \Delta$ *be a status value. The status of an intentional element* $e \in IE$ *is defined by the function* $\Phi : IE \to \Delta$*, where* $\Phi(e) = (d, d)$ *if* $e \in G \cup T$*, and* $\Phi(e) = d$ *if* $e \in Q$*.*

Goals and tasks take states $(d, d) \in \Delta \times \Delta$, where $d = (\top, \top)$ marks them as achieved but pending and $d = (\top, \bot)$ means achieved and not-pending. Qualities use a single $d \in \Delta$, where $d = \top$ means satisfied and $d = \top$ indicates that it is denied. Initially, all goals and tasks are marked as unknown with state $(?, ?)$, and all qualities are set to unknown $(?)$ (Figure 2). A goal model represents and tracks the state of its intentional elements.

Definition 6 (Goal Model). *Let* $\mathcal{MN} = \langle \mathcal{M}, alph, excluded, step \rangle$ *be an abstract model notation. Let* $GE = \langle IE, L \rangle$*, with* $IE = G \cup T \cup Q$ *and* $L = R \cup C$ *be the set of goal model elements (Definition 4). Let* $\Delta = \{\top, \bot, ?\}$ *be the set of truth-values and* $\Phi : IE \to \Delta$ *the status function of an intentional element (Definition5). Let* $GM \in \mathcal{M}$ *be a model instance. A goal model is defined as the tuple* $GM \triangleq \langle GE, ID_{GM}, iden_{GM} \rangle$*, where* GE *is a finite set of* goal model elements*,* ID_{GM} *is a finite set of identifiers, and* $iden_{GM} \subseteq IE \to ID_{GM}$ *is a bijection that assigns each intentional element a distinct identifier.*

Figure 2 shows a goal model based on the behaviour described in Section 2.1, with one *quality* ( ), two *goals* ( ), nine *tasks* ( ), refinements of type *and* ( ) and *or* ( ), and contribution links *Make* and *Break*. Each intentional

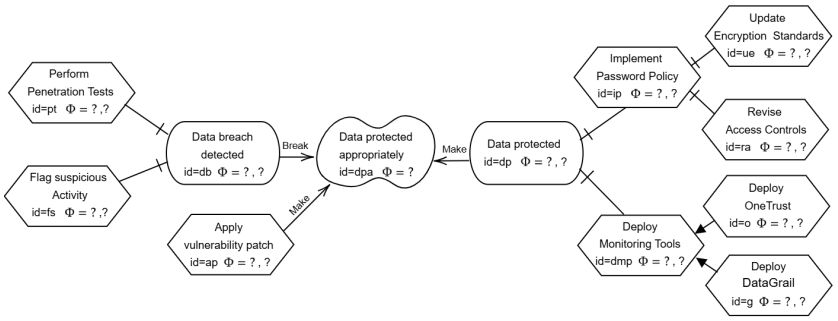


Fig. 2: A goal model GM, modeling the behaviour described in Section 2.1

element shows its identifier (id) and its status (Φ). A *goal model marking* μ^{GM} represents the current distribution of intentional elements' status. A *marked goal model* is a tuple of a goal model $GM = \langle GE, ID_{GM}, iden_{GM} \rangle$ and a goal model marking $\mu^{GM} \subseteq IE \to \Phi$, denoted by $\langle GM, \mu^{GM} \rangle$. For instance, the goal model marking in Figure 2 is $\langle$dpa :?, dp : (?, ?), ap : (?, ?), db : (?, ?), ip : (?, ?), dmp : (?, ?), ue : (?, ?), ra : (?, ?), o : (?, ?), g : (?, ?), fs : (?, ?), pt : (?, ?)$\rangle$.

Let $GE = \langle IE, L \rangle$ be the set of goal model elements, with $IE = G \cup T, \cup Q$ and $L = R \cup C$, and $e \in G \cup T$. The refinement reachability relation $\hookrightarrow \subseteq IE \times IE$ is defined as the smallest relation such that $(e, e') \in R$ implies $e' \hookrightarrow e$ (*direct*), and if $e' \hookrightarrow e$ and $e'' \hookrightarrow e'$, then $e'' \hookrightarrow^* e$ (*transitive*). Its reflexive–transitive closure $\hookrightarrow^*$ expresses reachability through zero or more refinement steps. For example, in Figure 2, the goal "*Data Protected*" (e) is refined through "*Deploy monitoring tools*" (e'), which includes tasks such as "*Deploy OneTrust*" (e''); thus, $e'' \hookrightarrow^* e$. To ensure analyzability for HL-BR satisfaction, we define well-formedness criteria constraining the structure and labeling of goal models.

Definition 7 (Well-formed Goal Model). *Let $GM = \langle GE, ID_{GM}, iden_{GM} \rangle$ be a goal model (Definition 6). Let $GE = \langle IE, L \rangle$, with $IE = G \cup T \cup Q$ and $L = R \cup C$ be the set of goal model elements (Definition 4). GM is* well-formed *iff: (1) GE is acyclic; (2) R and C are asymmetric; (3) There exist a minimum amount of goal model elements $|G \cup T| \geq 1$, $|Q| \geq 1$, and $\exists e \in G \cup T, q \in Q : C(e, q) =$ `Make`; (4) All refinements of e are the same type $\forall e \in G \cup T, \forall e_1, e_2 : (e, e_1), (e, e_2) \in R \Rightarrow R(e, e_1) = R(e, e_2)$; (5) Between any two intentional elements, at most one refinement link exists $\forall e_1, e_2 \in G \cup T : |\{(e_1, e_2) \in R\}| \leq 1$; (6) elements that contribute conflicting values to a quality must stem from disjoint refinement branches$\forall q \in Q, (e_1, q), (e_2, q) \in C, C(e_1, q) \neq C(e_2, q) : \{x \mid x \hookrightarrow^* e_1\} \cap \{y \mid y \hookrightarrow^* e_2\} = \emptyset$.*

Figure 2 depicts a well-formed goal model. We assume that every GM provided as input to our algorithm is well-formed. Based on this structure, we define its LTS, where states (S_{GM}) correspond to configurations reachable from the initial marking μ_0^{GM} via the transition relation $\to_{GM}$. We use the shorthand

$GM \models \mu_1^{GM} \xrightarrow{x} \mu_2^{GM}$ to denote the transition relation where GM does not change, that is $(\langle GM, \mu_1^{GM}\rangle, x, \langle GM, \mu_2^{GM}\rangle) \in \rightarrow_M$.

Definition 8 (Labeled Transition System Goal Model). *Let $\langle GM, \mu^{GM}\rangle$ be a marked goal model. Let $GM = \langle GE, ID_{GM}, iden_{GM}\rangle$ be a goal model (Definition 6). Let $GE = \langle IE, L\rangle$, with $IE = G \cup T \cup Q$ and $L = R \cup C$ be the set of goal model elements (Definition 4) Let $LTS_M = \langle S_M, Act_M, \longrightarrow_M, s_0^M, F_M\rangle$ be a labeled transition system of a model M (Definition 2). Let GM be the model M. A labeled transition system for a goal model is the tuple: $LTS_{GM} \triangleq \langle S_{GM}, Act_{GM}, \rightarrow_{GM}, s_0^{GM}, F_{GM}\rangle$, where:*

1. *S_{GM} is the set of states,*
2. *$Act_{GM} = G \cup T$ is the set of actions,*
3. *$\rightarrow_{GM} \subseteq S_{GM} \times Act_{GM} \times S_{GM}$, is the transition relation governed by the rules described in Figure 3,*
4. *$s_0^{GM} = \mu_0^{GM}$ is the initial state, s.t. $\mu_0^{GM}(e) = (?, ?)\ \forall e \mid e \in G \cup T$ and $\mu_0^{GM}(e) =? \ \forall e \mid e \in Q$,*
5. *$F_{GM} \subseteq S_{GM}$ are the final states s.t. $\forall q \in Q, \mu^{GM}(q) = \top$.*

Let $LTS_{GM} = \langle S_{GM}, Act_{GM}, \rightarrow_{GM}, s_0^{GM}, F_{GM}\rangle$ be a labeled transition system of a goal model G, (Definition 8). The rules (Figure 3) that govern the transition relation $\rightarrow_{GM}$ can be grouped in four categories: (1) *Activation*: A leaf node may fire if it has no refinements, (2) *Refinement Propagation*: If the conditions for the refinement type (i.e., *and*, *or*) are satisfied, then the element e in a refinement link (e, e') is marked as satisfied, (3) *Contribution Propagation*: If an element e with a contribution link to a quality q is satisfied, the status value of q will change according with the contribution type, (4) *Backpropagation*: As the satisfaction of the qualities evolve overtime, its effect is propagated backwards, indicating which executions are pending (i.e., need to be redone).

Let $GE = \langle IE, L\rangle$, where $IE = G \cup T \cup Q$ and $L = R \cup C$ be the set of goal model elements (Definition 4). Let $GM = \langle GE, ID_{GM}, iden_{GM}\rangle$ be the goal model (Definition 6). We write $GM = \langle G, T, Q, R, C, ID_{GM}, iden_{GM}\rangle$. Given a marking μ^{GM}, Figure 4 shows some of the reachable markings of the goal model shown in Figure 2. Each arrow is labeled with the identifier of the executed element and the transition rule applied for the resulting marking.

Lemma 1 (Goal Model as Abstract Model Notation). *Let $\mathcal{MN} = \langle \mathcal{M}, alph, excluded, step\rangle$ be an abstract model notation (Definition 1). Let $GE = \langle IE, L\rangle$, where $IE = G \cup T \cup Q$ and $L = R \cup C$ be the set of goal model elements (Definition 4). Let $GM = \langle GE, ID_{GM}, iden_{GM}\rangle$ be the goal model (Definition 6), and $\Phi(e)$ the status of $e \in G \cup T \cup Q$ (Definition 5). Let $\mathcal{M}$ be the set of all such GM where $\mu^{GM} : G \cup T \cup Q \rightarrow \Phi$, $\lambda = iden$ the labeling function, $excluded(GM) = \emptyset$, and $step \subseteq \mathcal{M} \times (G \cup T) \times \mathcal{M}$ where $(GM, e, GM') \in step$ iff e is executed according to Figure 3. Then $\mathcal{A} = \langle \mathcal{M}, \lambda, excluded, step\rangle$ is an abstract model notation.*

3.3 Process Models

A process model captures executable workflow steps; they can be modeled using imperative or declarative languages. Here, we use a subclass of Petri nets [25],

$$(\mathbf{P_{ie}}) \quad \frac{e \in G \cup T \quad e \vdash \text{leafnode}}{\langle G,T,Q,R,C,ID_{GM},iden_{GM}\rangle \models \mu^{GM} \xrightarrow{e} \mu^{GM}[e \mapsto (\top,\bot)]}$$

$$(\mathbf{P_{and}}) \quad \frac{(e,e') \in R \quad R(e,e') = \text{AND} \quad \forall e' \colon \mu^{GM}(e') = (\top,\bot)}{\langle G,T,Q,R,C,ID_{GM},iden_{GM}\rangle \models \mu^{GM} \xrightarrow{e} \mu^{GM}[e \mapsto (\top,\bot)]}$$

$$(\mathbf{P_{or}}) \quad \frac{(e,e') \in R \quad R(e,e') = \text{OR} \quad \exists e' \colon \mu^{GM}(e') = (\top,\bot)}{\langle G,T,Q,R,C,ID_{GM},iden_{GM}\rangle \models \mu^{GM} \xrightarrow{e} \mu^{GM}[e \mapsto (\top,\bot)]}$$

$$(\mathbf{P_{Make}}) \quad \frac{(e,e') \in C \quad e \in Q \quad \mu^{GM}(e) \in \{\top,?\} \quad \exists e' \colon \mu^{GM}(e') = \top \quad C(e,e') = \text{Make}}{\langle G,T,Q,R,C,ID_{GM},iden_{GM}\rangle \models \mu^{GM} \xrightarrow{e} \mu^{GM}[e \mapsto \top]}$$

$$(\mathbf{P_{Break}}) \quad \frac{(e,e') \in C \quad e \in Q \quad \mu^{GM}(e) \in \{\bot,?\} \quad \exists e' \colon \mu^{GM}(e') = (\top,\bot) \quad C(e,e') = \text{Break}}{\langle G,T,Q,R,C,ID_{GM},iden_{GM}\rangle \models \mu^{GM} \xrightarrow{e} \mu^{GM}[e \mapsto \bot]}$$

$$(\mathbf{BP_{fulfill}}) \quad \frac{(e,e') \in C \quad e \in Q \quad \mu^{GM}(e) = \bot \quad \mu^{GM}(e') = (\top,\bot) \quad C(e,e') = \text{Make}}{\langle G,T,Q,R,C,ID_{GM},iden_{GM}\rangle \models \mu^{GM} \xrightarrow{e} \mu^{GM\prime}}$$

where $\mu^{GM\prime} = \mu^{GM}[e \mapsto \top,\ e' \mapsto (\top,\top) \text{ for all } C(e,e') = \text{Break},\ e'' \mapsto (\top,\top) \text{ for all } e'' \hookrightarrow_R^* e']$

$$(\mathbf{BP_{deny}}) \quad \frac{(e,e') \in C \quad e \in Q \quad \mu^{GM}(e) = \top \quad \mu^{GM}(e') = (\top,\bot) \quad C(e,e') = \text{Break}}{\langle G,T,Q,R,C,ID_{GM},iden_{GM}\rangle \models \mu^{GM} \xrightarrow{e} \mu^{GM\prime}}$$

where $\mu^{GM\prime} = \mu^{GM}[e \mapsto \bot,\ e' \mapsto (\top,\top) \text{ for all } C(e,e') = \text{Make},\ e'' \mapsto (\top,\top) \text{ for all } e'' \hookrightarrow_R^* e']$

Fig. 3: Operational semantics of a goal model by extension $GM = \langle G,T,Q,R,C,ID_{GM},iden_{GM}\rangle$ under marking μ^{GM}, denoted as $GM \models \mu^{GM}$. Transition rules update the marking based on activation, refinement, and contribution.

called Workflow nets [1]. Let $\mathcal{MN} = \langle \mathcal{M}, alph, excluded, step\rangle$ be an abstract model notation (Definition 1). Let a Petri net N be a model instance $N \in \mathcal{M}$.

Definition 9 (Petri net). *(taken from [2]) A* Petri net *is a tuple* $N \triangleq \langle P,Tr,F, ID_{PM},iden_{PM}\rangle$ *where (1)* P *is a finite set of* places*, (2)* Tr *is a finite set of* transitions *such that* $P \cap Tr = \emptyset$*, (3)* $F \subseteq (P \times Tr) \cup (Tr \times P)$ *is a set of directed arcs, called the* flow relation*. A Petri net node is an element of* $P \cup Tr$*. (4)* ID_{PM} *is a finite set of unique* transition identifiers*, and (5)* $iden_{PM} : Tr \to ID_{PM}$ *is a* bijection *that assigns a distinct identifier to each transition.*

Figure 5a shows a Petri net, with ten places and eleven transitions. Token resides in places; p_0 has one token. A marking (μ^{PM}) represents token distribution. A *marked Petri net* is a tuple of a net $N = \langle P,Tr,F,ID_{PM},iden_{PM}\rangle$ and a marking $\mu^{PM} \in \mathbb{B}(P)$. The initial marking in Figure 5a is $[p_0]$; the set of all marked Petri nets is $\mathcal{N}$. A node x is called an *preset* of node y if $(x,y) \in F$, and an *postset* of y if $(y,x) \in F$. For example, ${}^\bullet t_4 = \{p_3,p_4\}$ and $t_4^\bullet = \{p_5\}$. A transition is enabled if all input places have tokens. Firing removes tokens from input places and adds tokens to output places.

Definition 10 (Firing Rule). *(taken from [2]) Let* $\langle N, \mu^{PM}\rangle$ *be a marked Petri net, with* $N = \langle P,Tr,F,ID_{PM},iden_{PM}\rangle$ *(Definition 9). A transition*

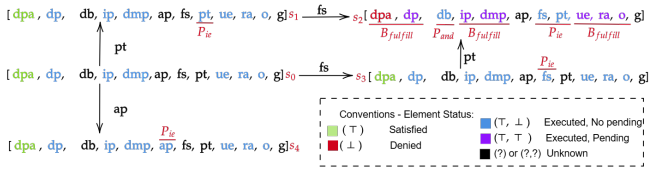


Fig. 4: Execution fragment of the LTS associated with the goal model in Fig. 6.

$t \in Tr$ *is* enabled*, denoted* $\langle N, \mu^{PM}\rangle[t]$*, if and only if* $\bullet t \leq \mu^{PM}$*. The firing rule* $[-] \subseteq \mathcal{N} \times Tr \times \mathcal{N}$ *is the smallest relation satisfying: for any* $\langle N, \mu^{PM}\rangle \in \mathcal{N}$ *and any* $t \in Tr$*,* $\langle N, \mu^{PM}\rangle[t] \Rightarrow \langle N, \mu^{PM}\rangle[t]\ \langle N, \langle \mu^{PM} \setminus \bullet t\rangle \cup t\bullet\rangle$

In the marking of Figure 5a, transition t_1 is enabled. Firing t_1 yields $[p_1, p_2]$. Transition sequences determine which markings are reachable and describe the dynamic behavior of the net.

Definition 11 (Firing Sequence). *(taken from [2]) Let* $\langle N, \mu^{PM}\rangle$ *be a marked petri net, with* $N = \langle P, Tr, F, ID_{PM}, iden_{PM}\rangle$ *(Definition 9). A sequence* $\sigma \in Tr^*$ *is a* firing sequence *of* $\langle N, \mu_0^{PM}\rangle$ *if, there exist markings* $\mu_1^{PM}, \ldots, \mu_n^{PM}$ *and transitions* $t_1, \ldots, t_n \in Tr \mid n \in \mathbb{N}$ *such that* $\sigma = \langle t_1 \ldots t_n\rangle$*, and for all* i *with* $1 \leq i < n$*, it holds that* $\langle N, \mu_i^{PM}\rangle[t_{i+1}]$ *and* $\langle N, \mu_i^{PM}\rangle[t_{i+1}] \rightarrow \langle N, \mu_{i+1}^{PM}\rangle$.

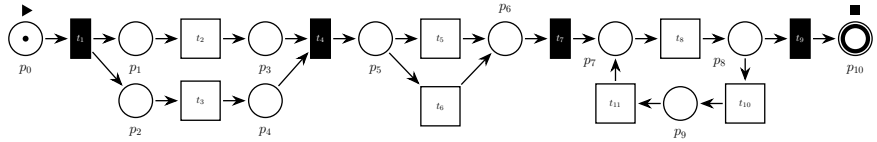
A marking μ^{PM} is *reachable* from μ_0^{PM} if a sequence of enabled transitions leads from μ_0^{PM} to μ^{PM}. Given an initial marking μ_0^{PM}, the set of reachable markings of $\langle N, \mu_0^{PM}\rangle$ is denoted S_{PM}, and can be computed using a *reachability graph* which is a specific type of labeled transition system.

Definition 12 (Labeled Transition System - Petri Net). *Let* $\langle N, \mu^{PM}\rangle$ *be the marked Petri net, with* $N = \langle P, Tr, F, ID_{PM}, iden_{PM}\rangle$ *(Definition 9. Let* N *be a model instance* M*. Let* $LTS_M = \langle S_M, Act_M, \longrightarrow_M, s_0^M, F_M\rangle$ *be a labeled transition system of a model* M *(Definition 2). A labeled transition system for a Petri net is the tuple:* $LTS_{PM} \triangleq \langle S_{PM}, Act_{PM}, \rightarrow_{PM}, s_0^{PM}, F_{PM}\rangle$ *where:*

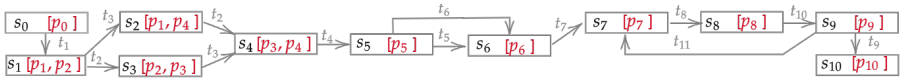
1. S_{PM} *is the set of states,*
2. $Act_{PM} = Tr$ *is the set of actions,*
3. $\rightarrow_{PM} \subseteq S_{PM} \times Tr \times S_{PM}$ *is the transition relation, it follows the firing rule (Definition 10), i.e.,* $(\mu^{PM}, t, \mu^{PM'}) \in \rightarrow_{PM}$ *iff* $\langle N, \mu^{PM}\rangle[t] \rightarrow \langle N, \mu^{PM'}\rangle$*,*
4. $s_0^{PM} = \mu_0^{PM}$*, is the initial state.*
5. $F_{PM} \subseteq S_{PM}$ *are the final states.*

Figure 5b shows the LTS of the Petri net in Figure 5a. We use Workflow nets to model processes with clear start, end, and control flow, with a unique entry point, a unique exit point, and well-defined execution semantics.

Definition 13 (Workflow nets). *(Taken from [2]) Let* $N = \langle P, Tr, F, ID_{PM}, iden_{PM}\rangle$ *be a Petri net (Definition 9) and* $\bar{t}$ *a fresh identifier not in* $P \cup Tr$*.* N *is a* workflow net (WF-net) *iff: (1)* P *contains an input place* i *s.t.* $\bullet i = \emptyset$*, (2)* P *contains an output place* o *s.t.* $o\bullet = \emptyset$*, and (3)* $\bar{N} = (P, Tr \cup \{\bar{t}\}, F \cup \{(o, \bar{t}), (\bar{t}, i)\})$ *is* strongly connected.



(a) a



(b) b

Fig. 5: A WF-net and its behaviour (a) A WF-net representing the example in Section 2.1; (b) visualizes its LTS. For example, t_2 represents *Update Encryption Standards*, t_3 *Revise the Access Controls*, t_5 and t_6 correspond to Deploy *DataGrail* or *OneTrust*, respectively, while t_8, t_{10} and t_{11} denote *Perform Penetration Tests*, *Flag suspicious activity* and *Apply Vulnerabilities Patches*.

Definition 14 (Safeness). *Let $WF = \langle N, \mu^{PM} \rangle$ be a marked Workflow net, with $N = \langle P, Tr, F, ID_{PM}, iden_{PM} \rangle$ (Definition 9). We say WF is safe if, for every reachable marking μ^{PM} from the initial marking μ_0^{PM}, it holds that $\mu^{PM}(p) \leq 1$ for all $p \in P$.*

Let $LTS_{PM} = \langle S_{PM}, Act_{PM}, \rightarrow_{PM}, s_0^{PM}, F_{PM} \rangle$ be the labeled transition system of a safe Workflow net, here $s_0^{PM} = \mu_0^{PM}$ with $\mu_0^{PM}(\blacktriangle) = 1$ and $\mu_0^{PM}(p) = 0$ for all $p \neq \blacktriangle$, and F_{PM} includes all markings with $\mu^{PM}(\blacksquare) = 1$ and $\mu^{PM}(p) = 0$ for all $p \neq \blacksquare$. Figure 5 depicts a safe Workflow net and its LTS. We assume that every process model used as input to our algorithm is a safe WF-net.

Lemma 2 (A Marked Workflow Net as Abstract Model Notation). *Let $MWN = \langle P, Tr, F, ID_{PM}, iden_{PM}, \mu_0^{PM} \rangle$ be a marked Workflow net (Definition 13). Define $\mathcal{M} = \left\{ \mu^{PM} : P \to \mathbb{N} \mid \exists \text{ firing sequence } \mu_0^{PM} \xrightarrow{t_1} \cdots \xrightarrow{t_n} \mu^{PM} \right\}$ as the set of reachable markings. Let $\lambda : \mathcal{M} \to 2^{ID_{PM}}$ be the labeling function defined by $\lambda(\mu^{PM}) = \left\{ iden_{PM}(t) \mid t \in Tr \text{ and } \forall p \in {\bullet t} : \mu^{PM}(p) \geq 1 \right\}$. Let excluded $: \mathcal{M} \to 2^{\emptyset}$ denote the exclusion function, and let step $\subseteq \mathcal{M} \times Tr \times \mathcal{M}$ be the transition relation such that $(\mu^{PM}, t, \mu'^{PM}) \in$ step if and only if $\mu^{PM} \xrightarrow{t} \mu'^{PM}$ holds according to the firing rule (Definition 10). Then $\mathcal{A} = \langle \mathcal{M}, \lambda, \text{excluded}, \text{step} \rangle$ is an abstract model notation (Definition 1).*

3.4 Mapping

To enable synchronization, we map process actions (transitions) to goal model elements (goals and tasks).

Definition 15 (Mapping between Process transitions and Intentional Elements). *Let $GM = \langle GE, ID_{GM}, iden_{GM} \rangle$ be a well-formed goal model. Let*

$N = \langle P, Tr, F, ID_{PM}, iden_{PM} \rangle$ be a safe Workflow net (Definition 13). The process transition to intentional element mapping is a total function defined as: map: $ID^{Tr}_{PM} \rightarrow ID^{G \cup T}_{GM} \cup \{\epsilon\}$ where $ID^{Tr}_{PM} \subseteq ID_{PM}$ is the set of identifiers assigned to process transitions, $ID^{G \cup T}_{GM} \subseteq ID_{GM}$ is the set of identifiers assigned to goals and tasks only, and ϵ denotes that a transition is left unmapped.

For example, consider the well-formed goal model in Figure 2 and the safe Workflow net shown in Figure 5a. Transitions $t_2, t_3, t_5, t_6, t_7, t_{10}, t_{11}$ are mapped to intentional elements with identifiers `ue` (*Update Encryption Standards*), `ra` (*Revise Access Controls*), `o` (*Deploy OneTrust*), `g` (*Deploy DataGrail*), `pt` (*Perform Penetration Tests*), `fs` (*Flag Suspicious Activity*), and `ap` (*Apply vulnerability patch*), respectively; all others (i.e., t_1, t_4, t_9) remain unmapped (ϵ). To ensure consistency, we introduce well-formedness criteria.

Definition 16 (Well-formed Mapping). *Let map : $ID^{Tr}_{PM} \rightarrow ID^{G \cup T}_{GM} \cup \{\epsilon\}$ be the process transition to intentional element mapping (Definition 15). Let $GE = \langle IE, L \rangle$, with $IE = \langle G \cup T \cup Q \rangle$ and $L = \langle R \cup C \rangle$ be the set of goal model elements (Definition 4). We say that map is* well-formed *if the following conditions holds:*

1. *Consistent contribution: Let $id \in ID^{Tr}_{PM}$ such that $map(id) = \{e_i, e_j, \dots\}$ and $|map(id)| \geq 2$. Let $e_i, e_j \in ID^{G \cup T}_{GM}$. For all intentional elements e_1, e_2 with identifier e_i, e_j and the same quality $q \in Q$:*
   (a) if e_1 or one of its descendants ($\forall e'$ s.t. $e' \hookrightarrow^* e_1$) contributes to q,
   (b) if e_2 or one of its descendants ($\forall e'$ s.t. $e' \hookrightarrow^* e_2$) also contributes to q,
   *(c) then e_1 and e_2 must contribute the same type (either **`Make`** or **`Break`**).*
   Formally, if there exist $e'_1, e'_2 \in G \cup T$ s.t. $e_1 = e'_1$ or $e_1 \hookrightarrow^* e'_1, e_2 = e'_2$ or $e_2 \hookrightarrow^* e'_2$ and $(e'_1, q), (e'_2, q) \in C$ then $C(e'_1, q) = C(e'_2, q)$
2. No and-refinement co-mapping*: For any process transition identifier $id \in ID^{Tr}_{PM}$, let $map(id) = \{e_i, e_j, \dots\} \subseteq ID^{G \cup T}_{GM}$. For all $e_i, e_j \in map(id)$, if there exist $e_1, e_2 \in G \cup T$ such that $iden_{GM}(e_1) = e_i$, $iden_{GM}(e_2) = e_j$, and there exists $R(e_1, e'_2) =$ and, and $e_2 \hookrightarrow^* e_1$ or $e_1 \hookrightarrow^* e_2$, then e_1 and e_2 can not be mapped by the same transition identifier.*

For example, in the goal model shown in Figure 2, a single process transition must not map to both the identifier `db` (*Data breach detected*) and `ap` (*Apply vulnerability patch*), since they contribute differently, one as `Make`, the other as `Break`, to the same quality. Similarly, a transition must not be mapped to both `ue` (*Update encryption standard*) and `ip` (*Implement password policy*), because there is an `and`-type refinement relation between them. We assume that every mapping used as input of our algorithm is well-formed.

4 Compliance Assessment of High-Level Business Requirements

We define compliance as the satisfaction of all high-level business requirements (HL-BR). HL-BR are represented as qualities in the goal model (Figure 2).

This section introduces the method to assess compliance at **design time** by evaluating whether the execution of process actions leads to a system state in which all the HL-BR are fulfilled. We introduce the synchronous product of the labeled transition system of each model and the compliance criterion.

Definition 17 (Composed Labeled Transition System). *Let* $LTS_{GM} = \langle S_{GM}, Act_{GM}, \rightarrow_{GM}, s_0^{GM}, F_{GM} \rangle$ *be the labeled transition system of the goal model (Definition 8). Let* $LTS_{PM} = \langle S_{PM}, Act_{PM}, \rightarrow_{PM}, s_0^{PM}, F_{PM} \rangle$ *be the labeled transition system of the process model (Definition 12). Their synchronous product is defined as:* $LTS_C \triangleq \langle S_C, Act_C, \rightarrow_C, s_0^C, F_C \rangle$ *where:*

1. $S_C = S_{GM} \times S_{PM}$ *is the set of composed states,*
2. $Act_C = Act_{GM} \cup Act_{PM} \cup \{\epsilon\}$ *is the set of actions, where* ϵ *represent a no observable action,*
3. $\rightarrow_C$ *is the transition relation, preserving synchronization of shared actions,*
4. $s_0^C = \langle s_0^{GM}, s_0^{PM} \rangle$ *is the initial state,*
5. $F_C = \{\langle s_{GM}, s_{PM} \rangle \in S_C \mid \forall q \in Q : \mu^{GM}(q) = \top\}$ *is the set of final states where all qualities are satisfied.*

Definition 18 (Compliance Criterion). *Let* $LTS_{GM} = \langle S_{GM}, Act_{GM}, \rightarrow_{GM}, s_0^{GM}, F_{GM} \rangle$ *be the LTS of a well-formed goal model (Definition 8). Let* $LTS_{PM} = \langle S_{PM}, Act_{PM}, \rightarrow_{PM}, s_0^{PM}, F_{PM} \rangle$ *be the LTS of a safe Workflow net (Definition 12). Let* $LTS_C = \langle S_C, Act_C, \rightarrow_C, s_0^C, F_C \rangle$ *be the synchronous product of* LTS_{PM} *and* LTS_{GM} *(Definition 17). The compliance satisfaction is defined as true if* $\forall s \in S_C,\ \exists s' \in F_C$ *such that* $s \rightarrow^* s'$*, and false otherwise.*

Let $GM = \langle GE, ID_{GM}, iden_{GM} \rangle$ be a well-formed goal model (Definition 6) Let $GE = \langle IE, L \rangle$, with $IE = G \cup T \cup Q$ and $L = R \cup C$ be the set of goal model elements (Definition 4). Let $N = \langle P, Tr, F, ID_{PM}, iden_{PM} \rangle$ be a safe Workflow net (Definition 13). Let $LTS_C = \langle S_C, Act_C, \rightarrow_C, s_0^C, F_C \rangle$ be the composed labeled transition system. Given an goal model marking μ^{GM} and process model marking μ^{PM} such that $GM \models \mu^{GM}$, and $N \models \mu^{PM}$. The operational semantics of the transition relation $\rightarrow_C$, is defined by the following rules:

$$\textbf{(P}_{\textbf{sync}}\textbf{)} \quad \frac{\begin{array}{c} map(iden_{PM}(t)) = iden_{GM}(e) \\ \langle G, T, Q, R, C, ID_{GM}, iden_{GM} \rangle \models \mu^{GM} \xrightarrow{e}_{GM} \mu^{GM'} \\ \langle P, Tr, F, ID_{PM}, iden_{PM} \rangle \models \mu^{PM} \xrightarrow{t}_{PM} \mu^{PM'} \end{array}}{\left\langle \begin{array}{l} G, T, Q, R, C, ID_{GM}, iden_{GM}, \\ P, Tr, F, ID_{PM}, iden_{PM} \end{array} \right\rangle \models (\mu^{GM}, \mu^{PM}) \xrightarrow{t}_C (\mu^{GM'}, \mu^{PM'})}$$

$$\textbf{(P}_{\textbf{local}}\textbf{)} \quad \frac{\begin{array}{c} map(iden_{PM}(t)) = \epsilon \\ \langle P, Tr, F, ID_{PM}, iden_{PM} \rangle \models \mu^{PM} \xrightarrow{t}_{PM} \mu^{PM'} \end{array}}{\left\langle \begin{array}{l} G, T, Q, R, C, ID_{GM}, iden_{GM}, \\ P, Tr, F, ID_{PM}, iden_{PM} \end{array} \right\rangle \models (\mu^{GM}, \mu^{PM}) \xrightarrow{t}_C (\mu^{GM}, \mu^{PM'})}$$

Fig. 6: Transition rules LTS_C

The composed LTS runs the process and goal models in a unified step, so every process transition either triggers the matching goal model update or leads to no changes. For instance, Figure 7 depicts two exemplary execution fragments of the labeled transition system composed for the goal model shown in Figure 2 and the Workflow net in Figure 5a. In the left side (*PM is compliant with GM*). States s_9, s_{10}, and s_{11} satisfy all the qualities (*dpa*) defined in the goal model, and therefore belong to the set F_C. If every state in LTS_C that is not shown in the exemplary execution eventually leads to at least one of these states, then the process model depicted in Figure 5a is considered compliant with the goal model presented in Figure 2, which means that the data was protected appropriately.

In contrast, in the right side scenario (*PM is non-compliant with GM*) consider a variation in the transition t_5 labels shown in Figure 12, instead of *"Deploy OneTrust"*, now the company will "Deploy Grafana" (t_5^*). Previously, the mapping of that transition was associated with the intentional element identified as *"o"* in the goal model shown in Figure 2, that is $map(t_5) = o$, but now with the mapping would be $map(t_5^*) = \epsilon$, meaning that t_5^* is not mapped to any intentional element. Since at least one state in LTS_C does not eventually reach a state where all qualities are satisfied, the process model including t_5^* is not compliant with the goal model shown in Figure 2.

Having established the formal rules for process execution and goal satisfaction, we now present the complete compliance-checking algorithm. The algorithm first constructs the state space of the composed Labeled Transition System LTS_C (Definition17) by iteratively applying the transition rules. Its complexity is $O(n)$ where $n = M \times K$, with $M = |S_{GM}|$ denoting the number of states in the goal model and $K = |S_{PM}|$ denoting the number of states in the process model. In the second part, compliance is verified by ensuring that for every state $s \in S_C$ there exists at least one reachable final state $s' \in F_C$ ($s \rightarrow^* s'$) such that $\forall q \in Q,\ \mu_{s'}^{GM}(q) = \top$. Termination is guaranteed by the finiteness of LTS_C. Note that while LTS_{GM} is always finite, the finiteness of LTS_{PM} holds only under specific properties of the process model (Definition 14).

The algorithm is modular and extensible, allowing improvements or adaptations to be incorporated according to the specific application domain or process analysis need. For instance, it can be easily modified based on a set of traces that determine which of these are compliant with the goal model, and the mapping can be redesigned and validated by legal and process experts.

5 Prototypical Implementation

We implemented our algorithm tasking as input a process model pattern derived from Article 17 of the GDPR, which defines the data subject's right to erasure, also known as the right to be forgotten. This article contains high-level business requirements (HL-BR) such as *Data deleted without undue delay*, *when no longer necessary*, *after consent withdrawal*, and *Data retained* when *overriding legitimate grounds*, and *public interest*. The requirements were modeled in a goal model and refined into tasks. Each process activity was associated with

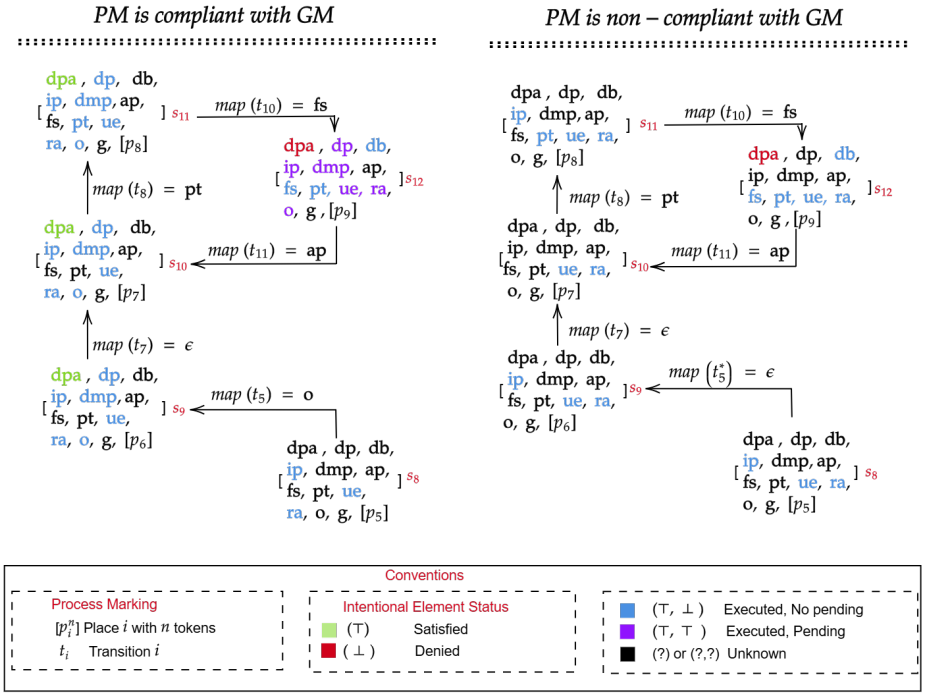


Fig. 7: Execution of the composed LTS of the Marked Workflow net in Figure 5a, and the goal model in Figure 6. The left side shows an example of a compliant execution, and the right side shows the counterexample of a non-compliant case.

some intentional element. Compliance was evaluated at design time by verifying the alignment between goal satisfaction and process execution behavior. We compared the capabilities of our framework against the requirements and interactions described in Art. 17 of the GDPR, such as the request for data erasure, verification of the data subject identity, evaluation of legal exceptions, and the actual deletion of personal data. These interactions serve as a representative example of the type of multi-step, conditional processes that are legally regulated. The evaluation was used to reason about the expressiveness of the framework. Our case study showed enough expressiveness to capture most requirements; however, requirements describing interprocess communication and relationships among multiple agents and their goals are still missing. Moreover, the current version does not yet support certain expressive constructs of process models, such as data conditions and timed events. The prototype implementing our approach is available in Python 3[1]. It requires three inputs to perform design-time compliance checking: a process model in PNML format, a goal model in JSON format, and a mapping file in CSV format.

[1] `https://github.com/jc4v1/HLBRBPM25`

Algorithm 1: Compliance Checking Algorithm

Input: Goal model $GM = \langle GE, ID_{GM}, iden_{GM} \rangle$,
Process model $PM = \langle P, Tr, F, ID_{PM}, iden_{PM} \rangle$,
Mapping function $map : ID_{PM}^{Tr} \rightarrow ID_{GM}^{G \cup T} \cup \{\epsilon\}$
Output: `true` if every state leads to one where all $q \in Q$ are satisfied; `false` otherwise

Initialize $S_C := \{\langle \mu_0^{GM}, \mu_0^{PM} \rangle\}$;
Construct state space by exploring all enabled transitions $t \in Tr$; for each, update μ^{PM} and apply goal model semantics to μ^{GM} if $map(iden_{PM}(t)) \neq \epsilon$; add resulting states to S_C;
foreach $s \in S_C$ **do**
    if *no reachable* $s' \in S_C$ *from* s *with* $\mu^{GM'}(q) = \top$ *for all* $q \in Q$ **then**
        return *false* // Not compliant
    end
end
return *true* // Compliant

6 Related Work

Recent advances in process compliance have emphasized the alignment of operational processes with regulatory and business constraints through formal methods such as conformance checking and logic-based validation. Foundational techniques focus on aligning observed executions with procedural models (i.e., Petri nets) or multi-perspective specifications that include data and resources [5, 6]. However, most approaches presuppose a complete, procedural specification and struggle with high-level requirements expressed in non-operational terms, such as stakeholder goals or soft constraints. While extensions using preferences [18] or alignment of goals and processes [15] offer expressive power, they lack runtime verification semantics or cannot systematically integrate with trace-level abstractions. As such, the space between high-level intent modeling and low-level event traces remains fragmented, limiting formal reasoning about compliance.

To address this, recent efforts have explored mapping business requirements to executable process behaviors, but largely from the perspective of conformance, not enforcement or design-time guarantee. For example, works like [4] investigate the conformance space by comparing different behavioral equivalences, yet stop short of offering a mechanism to ensure satisfaction of non-functional constraints during execution. Predictive approaches [27] further extend the vision by forecasting compliance outcomes, but do not explicitly link these to stakeholder-defined goals. Moreover, alignment-based conformance checking [17] focuses on decomposing data-aware models for efficiency, not for semantic integration with intentional structures. This gap shows an opportunity for frameworks that unify abstract goal models and operational models through shared transition semantics, enabling formal guarantees over business objectives at runtime.

7 Conclusion and Future Work

We proposed a design-time compliance framework that integrates Workflow Petri nets with i goal models using synchronized labeled transition systems to assess whether all high-level business requirements (HL-BR) are satisfied. The key

contributions include: (1) a formal operational semantics for goal models with support for conflict resolution; (2) a compliance-checking algorithm based on synchronous composition of process and goal models; and (3) a prototype implementation that evaluates compliance from model inputs. Current limitations include a lack of support for timed events and capturing full model-driven language expressiveness. Future work will address this by extending the framework to support time, data, and multi-agent behavior, and adding a runtime monitor for detecting violations during execution.

Acknowledgments. This work was supported by the research grant "Center for Digital ComplianceE (DICE)" (VIL57420) from VILLUM FONDEN, and by the Innovation Foundation project "Explainable Hybrid-AI for Computational Law and Accurate Legal Chatbots" 4355-00018B XHAILe.

References

1. van der Aalst, W.M.P.: The application of petri nets to workflow management. The Journal of Circuits, Systems and Computers **8**(1), 21–66 (1998)
2. Aalst, W.M.V.D., Dongen, B.F.V.: Discovering petri nets from event logs. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 7480 LNCS (2013)
3. Amyot, D., Akhigbe, O., Baslyman, M., Ghanavati, S., Ghasemi, M., Hassine, J., Lessard, L., Mussbacher, G., Shen, K., Yu, E.: Combining goal modelling with business process modelling: Two decades of experience with the user requirements notation standard. Enterprise Modelling and Information Systems Architectures **17** (2022)
4. Burattin, A., Gianola, A., López, H.A., Montali, M.: Exploring the conformance space. In: CEUR Workshop Proceedings. vol. 2952 (2021)
5. Burattin, A., Maggi, F.M., Sperduti, A.: Conformance checking based on multi-perspective declarative process models. Expert Systems with Applications **65** (2016)
6. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance checking: Relating processes and models, relating processes and models. Springer (2018)
7. Dalpiaz, F., Franch, X., Horkoff, J.: istar 2.0 language guide (2016)
8. Debois, S., López, H.A., Slaats, T., Andaloussi, A.A., Hildebrandt, T.T.: Chain of events: Modular process models for the law. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 12546 LNCS (2020)
9. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Information and Software Technology **50** (2008)
10. Franceschetti, M., Seiger, R., López, H.A., Burattin, A., García-Bañuelos, L., Weber, B.: A characterisation of ambiguity in BPM. In: LNCS. vol. 14320 LNCS (2023)
11. Franch, X., López, L., Cares, C., Colomer, D.: The i* framework for goal-oriented modeling, pp. 485–506. Springer International Publishing, Cham (2016)
12. Ghasemi, M., Amyot, D.: From event logs to goals: a systematic literature review of goal-oriented process mining. Requirements Engineering **25** (2020)

13. Governatori, G.: The regorous approach to process compliance. In: Proceedings of the 2015 IEEE 19th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations, EDOCW 2015 (2015)
14. Grisold, T., Mendling, J., Otto, M., vom Brocke, J.: Adoption, use and management of process mining in practice. Business Process Management Journal **27** (2021)
15. Guizzardi, R., Reis, A.N.: A method to align goals and business processes. In: Lecture Notes in Computer Science. vol. 9381 (2015)
16. Hildebrandt, T.T., Mukkamala, R.R.: Declarative event-based workflow as distributed dynamic condition response graphs. Electronic Proceedings in Theoretical Computer Science **69** (2011)
17. de Leoni, M., Munoz-Gama, J., Carmona, J., van der Aalst, W.M.: Decomposing alignment-based conformance checking of data-aware process models. In: Lecture Notes in Computer Science. vol. 8841 (2014)
18. Liaskos, S., McIlraith, S.A., Sohrabi, S., Mylopoulos, J.: Representing and reasoning about preferences in requirements engineering. Requirements Engineering **16** (2011)
19. López, H.A., Massacci, F., Zannone, N.: Goal-equivalent secure business process re-engineering. In: International Conference on Service-Oriented Computing. pp. 212–223. Springer (2007)
20. López, H.A.: Challenges in legal process discovery. In: CEUR Workshop Proceedings. vol. 2952 (2021)
21. López, H.A., Debois, S., Slaats, T., Hildebrandt, T.T.: Business process compliance using reference models of law. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 12076 LNCS (2020)
22. López, H.A., Hildebrandt, T.T.: Three decades of formal methods in business process compliance: A systematic literature review (2024)
23. Monciardini, D., Bernaz, N., Andhov, A.: The organizational dynamics of compliance with the uk modern slavery act in the food and tobacco sector. Business and Society **60** (2021)
24. Morandini, M., Penserini, L., Perini, A.: Operational semantics of goal models in adaptive agents. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS. vol. 1 (2009)
25. Reisig, W., Rozenberg, G. (eds.): Lectures on Petri Nets I: Basic Models, Lecture Notes in Computer Science, vol. 1491. Springer-Verlag, Berlin (1998)
26. Riemsdijk, M.B.V., Dastani, M., Winikoff, M.: Goals in agent systems: A unifying framework. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS. vol. 2 (2008)
27. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. ACM Trans. Knowl. Discov. Data **13**(2) (2019)
28. Varela-Vaca, A.J., Gómez-López, M.T., Morales Zamora, Y., M. Gasca, R.: Business process models and simulation to enable GDPR compliance. Int. J. Inf. Secur. (2025)
29. Yu, E., et al.: Modeling strategic relationships for process reengineering. Social Modeling for Requirements Engineering **11**(2011), 66–87 (2011)