On the Formal Semantics of the Extended
Compliance Rule Graph *

David Knuplesch¹, Manfred Reichert¹, Linh Thao Ly¹, Akhil Kumar², and
Stefanie Rinderle-Ma³

¹ Institute of Databases and Information Systems, Ulm University, Germany
david.knuplesch,manfred.reichert,thao.ly@uni-ulm.de
² Smeal College of Business, Pennsylvania State University, PA, USA
akhilkumar@psu.edu
³ Faculty of Computer Science, University of Vienna, Austria
stefanie.rinderle-ma@univie.ac.at

Abstract. A fundamental challenge for any process-aware information
system is to ensure compliance of modeled and executed business pro-
cesses with imposed compliance rules stemming from guidelines, stan-
dards and laws. Such compliance rules usually refer to multiple process
perspectives including control flow, data, time, and resources as well
as interactions with business partners. On one hand, compliance rules
should be comprehensible for domain experts who must define and ap-
ply them. On the other, compliance rules should have precise semantics
such that they can be automatically processed. In this context, providing
a visual compliance rule language, which hides formal details from rule
designers, is crucial in order to enable an intuitive way of modeling. So
far, visual compliance rule languages have focused on the control flow
perspective, but lack adequate support for the other perspectives. To
remedy this drawback, this report introduces the extended Compliance
Rule Graph language and its formal semantics.

Keywords: business process compliance, compliance rule graphs, busi-
ness process modeling, business intelligence, formal semantics

1 Introduction

During the last decade, numerous approaches for ensuring the correctness of
business processes have been discussed [2–5]. Most of them focus on syntac-
tical correctness and process model soundness (e.g., absence of deadlocks and
lifelocks). However, business processes must also comply with semantic rules
stemming from domain-specific requirements such as corporate standards, le-
gal regulations, guidelines or best practices [6, 7]. Summarized under the no-
tion of business process compliance, existing approaches have mostly consid-

* This work was done within the research project C³Pro funded by the German Re-
search Foundation (DFG), Project number: RE 1402/2-1, and the Austrian Science
Fund (FWF) under project number: I743. Parts of this technical report will be pub-
ished in [1].
compliance issues related to the control flow perspective of single processes. In turn, only a few approaches consider the data, resource, and time perspectives in this context, although they are crucial as well [8–11]. Furthermore, cross-organizational scenarios characterized by interacting and collaborating business processes of various parties have not been properly considered so far [12]. In this context, compliance requirements need to be specified for both local and global processes as well. Note that this requires to consider the data, resource, and time perspectives as well as interactions between business partners (i.e. messages exchanged) as well. As examples consider the compliance rules in Table 1, which are imposed on a cross-organizational process scenario involving the two business partners reseller and manufacturer. In particular, as shown by the highlighted terms in Table 1, compliance rules may refer to the data, time, and resource perspectives of business processes as well as to interactions with business partners.

Compliance rule $c_1$ refers to the interactions between a reseller and manufacturer (request and reply) after a particular point in time (3rd January, 2013) as well as the maximum time distance between them (within three days). In turn, the data perspective of compliance rules is emphasized by compliance rule $c_2$ of the manufacturer. It forbids changing an order after having started the corresponding production task. Compliance rule $c_3$ combines the interaction, time, and data perspectives. Finally, compliance rule $c_4$ introduces the resource perspective (member of the order processing department and another member of the same department with supervisor status). In addition, $c_4$ considers the data perspective (e.g. new customer and total amount greater than €5,000) and the time perspective (at most three days). Particularly, $c_4$ shows that the different perspectives might be relevant for the same rule and hence must not be considered in an isolated manner.

When comparing $c_4$ and $c_2$ with $c_1$ and $c_3$, one can observe two different viewpoints: $c_4$ and $c_2$ are expressed from the viewpoint of the manufacturer (i.e., local view), whereas $c_1$ and $c_3$ reflect a global view. Note that such distinction

<table>
<thead>
<tr>
<th>Compliance Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>Any request sent from the reseller to the manufacturer after January 3rd, 2013 should be replied by the manufacturer within three days.</td>
</tr>
<tr>
<td>$c_2$</td>
<td>After starting the production related to a particular order, the latter must not be changed anymore.</td>
</tr>
<tr>
<td>$c_3$</td>
<td>When the manufacturer sends a bill with an amount lower than €5,000 to the reseller, the latter must make the payment within 7 days.</td>
</tr>
<tr>
<td>$c_4$</td>
<td>After receiving a production request message from the reseller, which refers to a new customer and has a total amount greater than €5,000, the solvency of this customer must be checked by a member of the order processing department. Based on the result of this check, another member of the same department with supervisor status must approve the request. Finally, the approval result must be sent to the reseller at most three days after receiving the original request.</td>
</tr>
</tbody>
</table>
between local and global views is common to cross-organizational collaboration scenarios not only in the context of process compliance [12, 13]. For example, BPMN 2.0 provides collaboration and choreography diagrams to express these different viewpoints.

Several approaches for formally capturing compliance requirements at different abstraction levels (e.g., temporal logics [14]) exist. In particular, they also enable the automatic verification of the conformance of business processes with respective compliance rules. As the use of formal languages for specifying compliance rules would be too intricate, rule patterns hiding formal detail from rule modelers have been proposed [9, 11]. Furthermore, a few approaches also consider more advanced issues like, for example, the use of data conditions in the context of compliance requirements. However, existing approaches are usually restricted to a specific subset of rule patterns. In turn, languages employing visual notations, like the compliance rule graph approach [15] or BPSL [16], combine an intuitive notation with the advantages of a formal language. However, the meta-analyses and case studies, we conducted in domains like higher education [17], medicine [18–21] and automotive engineering [22, 23], have revealed that these visual compliance rule languages still lack support for the time, data, and resource perspective of business processes and do not consider cross-organizational scenarios with interacting partners [12]. Overall, the following fundamental requirements for visual compliance rule languages need to be considered:

- In addition to the control flow perspective, the data, resource and time perspectives of compliance requirements must be properly captured.
- To not only consider process orchestrations, but cross-organizational scenarios as well, it becomes necessary to integrate the interaction perspective with compliance rule languages as well.
- To provide tool support for both the modeling and verification of compliance rules, both their syntax and semantics must be formalized.

To cope with the discussed shortcomings, this report introduces the extended Compliance Rule Graph (eCRG) and its formal semantics. The eCRG builds on the Compliance Rule Graph (CRG) language developed in previous work [15, 24], but additionally comprises elements enabling the visual modeling of compliance rules with the support of the process, data, time, and resource perspectives. Furthermore, we introduce concepts that allow defining compliance rules in respect to message flows and partner interactions; i.e., the eCRG language is able to specify compliance requirements for cross-organizational process scenarios (i.e. processes choreographies). For defining the formal semantics of the eCRG, we provide a transformation into FOL formula. Note that the latter can be evaluated over process traces to a posteriori verify the compliance of business processes. Altogether, the eCRG allows capturing compliance requirements at an abstract level, while enabling the specification of verifiable compliance rules in the context of cross-organizational scenarios, including the process, data, time, and resource perspectives.
The remainder of this report is structured as follows: Section 2 discusses related work. Section 3 introduces the extended Compliance Rule Graph (eCRG). The formal semantics of the eCRG language is specified in Section 4, whereas Section 5 provides a pattern-based evaluation. Finally, Section 6 concludes the report and provides an outlook on future research.

2 Related Work

During the last years the interaction, time, resource, and data perspectives have been considered in business process modeling in addition to the control flow perspective (e.g., [25–33]). The integration of business process compliance throughout the entire process lifecycle has been discussed in [24, 34–36]; [37] examined compliance issues in the context of cross-organizational processes developing a logic-based formalism for describing the semantics of normative specifications as well as compliance checking procedures. This approach allows modeling business obligations regulating the execution of business processes. In turn, [38] introduced a semantic layer that interprets process instances according to an independently designed set of internal controls. Furthermore, there exist approaches using semantic annotations to ensure compliance [39]. An approach for checking the compliance of process models against semantic constraints as well as for ensuring the validity of process change operations based on Mixed-Integer Programming formulation is proposed in [40]. It introduces the notions of degree of compliance, validity of change operations, and compliance by compensation. Further, [9] uses alignments to detect compliance violations in process logs. To verify whether compliance rules are fulfilled by process models at design time, many approaches apply model checking techniques [14, 16, 41–44]; some of them address the data and time perspectives as well. In order ensure compliability and global compliance of cross-organizational processes, where partners only provide restricted views on their local processes, [45, 46] apply model checking as well. Other approaches for verifying compliance apply the notion of semantic congruence [47] or use petri-nets [48] and consider the data and time perspectives as well.

The approach described in [41, 49] for visually modeling compliance considers the control flow and data perspectives. It is based on linear temporal logic (LTL), which allows modeling the control flow perspective based on operators like next, eventually, always, and until. Other visual approaches for compliance rule modeling [15, 16, 50] focus on control flow and partially the data perspective, but ignore the other perspectives mentioned.

3 Extended Compliance Rule Graphs

This section introduces extended Compliance Rule Graphs (eCRG) – a visual notation for compliance rule modeling covering the process, interaction, time, resource, and data perspectives. Section 3.1 introduces fundamentals of CRGs. Its extensions are introduced stepwise in Sections 3.2–3.6.
3.1 Fundamentals of Compliance Rule Graphs

The compliance rule graph (CRG) language [15, 24] allows visually modeling compliance rules whose semantics is defined over event traces. More precisely, a CRG is an acyclic graph consisting of an antecedence pattern as well as at least one related consequence pattern. Both patterns are modeled using occurrence and absence nodes, which indicate the occurrence or absence of events (e.g., related to the execution of a particular task). Edges between such nodes indicate control flow dependencies. As illustrated in Fig. 1, a trace is considered as compliant with a CRG if for each match of the antecedence pattern there is at least one corresponding match of every consequence pattern. Furthermore, a trace is considered as trivially compliant if there is no match of the antecedence pattern. For example, the CRG from Fig. 1 expresses that for each B not preceded by an A, there must occur a D, which is not preceded by any C also preceding the corresponding B.

![Image of CRG example and semantics over execution traces](image)

In the following sections, we introduce the extended Compliance Rule Graph (eCRG) language, which is based on CRGs. In addition to using nodes and connectors (i.e., edges), eCRG allows for attachments. The latter represent constraints to the nodes or edges they are attached to. Furthermore, an eCRG may contain instance nodes representing particular instances, which exist independently from the respective rule (e.g. a particular employee Mr. Smith, date 3rd January 2013, or role supervisor). Accordingly, instance nodes are neither part of the antecedence nor the consequence pattern.

3.2 Process Perspective

We first consider the process (i.e., control flow) perspective. The eCRG elements for modeling the process (i.e. control flow) perspective of compliance rules are introduced in Fig. 2. Since the extensions are based on the CRG language, there exist four different task elements, i.e., antecedence occurrence, antecedence absence, consequence occurrence, and consequence absence tasks. These allow expressing whether or not particular tasks must be executed. In addition, two different kinds of sequence flow connectors are provided that may be used to constrain
the execution sequence of tasks. Note that the absence of sequence flow indicates a parallel flow. To clearly distinguish between start-start, start-end, end-start, and end-end constraints on the execution sequence of tasks, sequence flow edges are either connected to the right or left border of a task node. Furthermore, exclusive connectors allow modeling mutual excluding tasks. Alternative connectors, in turn, express that at least one of the connected tasks must occur. Note that exclusive as well as alternative connectors may only connect nodes that are either part of the antecedence or the consequence pattern.

Fig. 4A shows an example of a start-start constraint on the execution sequence of tasks. It depicts the process perspective of compliance rule $c_2$ from Table 1 that disallows executing task change order after task production is started.

3.3 Interaction Perspective

The interaction perspective of business processes is crucial in cross-organizational scenarios [51, 13]. It covers constraints on the messages exchanged between business partners and the interaction view of the eCRG meta-model. Message exchanges are expressed in terms of particular nodes that reflect the events of sending and receiving a message. In turn, a message flow denotes the dependency between the events representing the sending and receiving of a particular message (cf. Fig. 3).
In Fig. 4B, the elements from Fig. 3 are used to model the process and interaction perspective of compliance rule $c_4$. This rule requires that after receiving message request from a reseller, a solvency check must be performed first. Then, a decision about approval has to be made before replying the request. Although the rule modeled in Fig. 4B considers the interaction perspective, using the two message nodes request and reply, it still represents the view of a particular business partner on its local business processes. We refer to this traditional point of view as the local view of a compliance rule. However, when considering the compliance rules $c_1$ and $c_3$ from Table 1, one can easily discover a global point of view on cross-organizational processes and related interactions (i.e., the messages exchanged) that corresponds to the BPMN 2.0 choreography diagram. In this interaction view, interaction nodes (cf. Fig. 3) are used to denote the exchange of a message between two business partners. Since the interaction view spans multiple business partners, task nodes may be annotated with the the business partner responsible (cf. Fig. 2).

Fig. 5A provides an interaction view on compliance rule $c_1$ from Table 1: After the reseller sends a request to the manufacturer, eventually, the manufacturer must reply. Furthermore, Fig. 5B provides an interaction view on compliance rule $c_3$ from Table 1. This rule requires that the reseller must perform task payment after having received billing information from the manufacturer.

### 3.4 Time Perspective

The time perspective of a business process deals with temporal constraints that need to be obeyed by instances of the process [28, 29].

Having a closer look on the original definition of compliance rules $c_1$ and $c_3$ from Table 1, it becomes clear that the eCRGs from Figs. 5A and 5B do not fully specify them yet. In particular, the time distances between the interactions and tasks have not been modeled. Fig. 6 provides elements for modeling points in time and time conditions in compliance rules. The latter may be attached to task nodes as well as sequence or message flow connectors to either constrain the duration of a task or the time distance between tasks, messages or points in time. Additionally, time distance connectors are introduced that must be attached
with a time condition. Respective time distance connectors and related time conditions then allow constraining the time distance between tasks, messages or points in time without implying a particular sequence.

Fig. 7A combines the interaction and time perspectives of compliance rule \textit{c}_1. This visual representation of \textit{c}_1 covers exactly the semantics of the compliance rule described in Table 1. In Fig. 7B, the interaction and time perspectives of \textit{c}_3 are provided. This compliance rule requires that at most seven days after the manufacturer sends \textit{billing information} to the reseller, the latter must perform task \textit{payment}.

### 3.5 Resource Perspective

The resource perspective covers the different kinds of human resources as well as their inter-relations [25, 52]. Further, it allows constraining the assignment of resources to tasks. In the context of our work, we consider resources like \textit{staff member}, \textit{role}, \textit{group}, and \textit{organizational unit} as well as their relations to tasks. Furthermore, we support \textit{resource conditions} and \textit{relations} among resources (cf. Fig. 8). Similar to task nodes, \textit{resource nodes} may be part of the antecedence or consequence patterns. Alternatively, they may represent a particular resource instance (e.g. staff member \textit{Mr. Smith}, or role \textit{supervisor}). In turn, \textit{resource conditions} may constrain resource nodes. Furthermore, the \textit{performing relation} indicates the performer of a task. Finally, \textit{resource relation connectors} express relations between resources. Note that the resource perspective can be easily extended with other kinds of resources if required.

Fig. 9 combines the process, interaction, time, and resource perspectives of compliance rule \textit{c}_4. This rule requires that at least three days after receiving a \textit{request} of the reseller, a \textit{replay} must be sent to him. Before sending this reply, first of all, task \textit{solvency check} must be performed by a staff member assigned to the particular organizational unit \textit{order processing department}. Following this task, another staff member of the same department with \textit{supervisor} status (i.e., role) must decide whether to grant approval before sending the \textit{reply}.
3.6 Data Perspective

The data perspective of business processes covers the data objects processed as well as the data flow between the tasks of the process [32, 53, 33]. Fig. 8 introduces eCRG elements for modeling data containers and data objects as well as connectors representing data flow. Thereby, data containers refer to process data elements or global data stores. By contrast, data objects refer to particular data values and object instances. Similar to resource nodes, data nodes may be part of the antecedence or consequence pattern, or represent a particular data container or data object (e.g., data container student credit points, document 1st order from Mr. Smith). Furthermore, data flow defines which process tasks read or write which data objects or data container. To constrain data container, data objects, and data flow, data conditions may be attached. Finally, data relation connectors may be used either to compare different data objects or to constrain the value of data containers at particular points in time.

Figs. 10A-C show the visual modeling of compliance rules $c_2$, $c_3$, and $c_4$ covering the data perspective as well as the other perspectives discussed. Note that each of the depicted eCRG covers the informal semantics described in Table 1.
Fig. 10: Local view on c₂ and interaction view on c₃ and c₄, considering the process, interaction, time, resource, and data perspectives

4 Formal Semantics of the eCRG

In the previous section, we have only informally described eCRG semantics. In order to enable automated compliance checking of process logs against eCRG, however, an unambiguous formal semantics is required. For this purpose, this section provides a transformation from eCRG to first-order predicate logic (FOL), which defines how to interpret and evaluate an eCRG over process logs. How this transformation can be accomplished is described in the following. First, Section 4.1 introduces sets representing fundamental business process concepts (e.g., tasks types, resources, or data values). Second, process logs are formally defined as set of predicates in Section 4.2. Third, Section 4.3 provides a set-based formal description of an eCRG. Fourth, Section 4.4 defines the transformation translating the eCRG into logic formula based on the defined predicates.

4.1 Environmental Sets and Concepts

The eCRG language relates to different perspectives of business processes, i.e., the process, time, data, resource, and interaction perspectives. The formalization of the eCRG language requires a formal definition of theses fundamental concepts defining the process environment in Def. 1 as follows:

**Definition 1 (Process Environment).**

The environment of a business process comprises the following sets:

- $\mathcal{T}$ is the set of task types
- $\mathcal{M}$ is set of message types
- $\mathcal{S}$ is the discrete and ordered set of points in time
- $\mathcal{R} := \mathcal{R}_S \cup \mathcal{R}_R \cup \mathcal{R}_G \cup \mathcal{R}_U$ is the set of resources, with $\mathcal{R}_S$ being the set of staff members, $\mathcal{R}_R$ being the set of roles, $\mathcal{R}_G$ being the set of groups, $\mathcal{R}_U$ being the set of business units and departments.
- $\mathcal{D}$ set of data containers, $\mathcal{P}$ set of activity and message parameters,
- $\mathcal{Q}$ set of data values and objects (e.g. 5, 1,345, color red, document-A, report-123, ...),
- $I_T$ set of task instance identifiers, and
- $I_M$ set of message instance identifier.
Further, based on the definition of the environment, we introduce the following sets of conditions and relations for time, data and resources:

- $C_T \subseteq \{c_{\text{con}} | c_{\text{con}} \subseteq T \rightarrow \exists \}$ is the set of time conditions,
- $C_D \subseteq \{c_{\text{con}} | c_{\text{con}} \subseteq \Omega \rightarrow \exists \}$ is the set of data conditions,
- $C_R \subseteq \{c_{\text{con}} | c_{\text{con}} \subseteq \mathbb{R} \rightarrow \exists \}$ is the set of resource conditions,
- $R_T \subseteq \{r_{\text{rel}} | r_{\text{rel}} \subseteq T^2 \rightarrow \exists \}$ is the set of time relations,
- $R_D \subseteq \{r_{\text{rel}} | r_{\text{rel}} \subseteq \Omega^2 \rightarrow \exists \}$ is the set of data relations, and
- $R_R \subseteq \{r_{\text{rel}} | r_{\text{rel}} \subseteq \mathbb{R}^2 \rightarrow \exists \}$ is the set of resource relations.

InStaff $\in R_R$ is a resource relation and InStaff$(r, s)$ indicates that staff member $s$ belongs or is assigned to resource $r$. Furthermore, we do not explicitly distinguish between time conditions and time relations, since each time condition $c_{\text{con}}$ constitutes a time relation $r_{\text{rel}}$ with

$$r_{\text{rel}}(t_1, t_2) \iff c_{\text{con}}([t_1 - t_2]).$$

### 4.2 Process Logs

A common way to capture the execution of a business process is to store execution events in process logs and traces [2]. Usually, the latter contain multiple information about the tasks executed, messages exchanged, or data objects accessed. However, not all logged information is of interest for evaluating whether or not a particular process instance complies with a given compliance rule. Independent from the concrete format of a process log, we introduce a set of predicates describing its relevant information in Def. 2. Obviously, the values of these predicates can be easily derived by processing a concrete process log:

**Definition 2 (Execution Log Entries/Predicates).**

Let $T, M, \Sigma, \mathbb{P}, D, T^*, I^*$ be defined as in Def. 1. Then, predicate

- Start$(t, i, t_t)$ expresses that task $i \in T^*$ of type $t_t \in T$ is started at $t \in \Sigma$.
- End$(t, i, t_t)$ expresses that task $i \in T^*$ of type $t_t \in T$ is completed at $t \in \Sigma$.
- Perform$(t, i, s)$ expresses that task $i \in T^*$ was performed by the staff member $s \in \mathbb{R}$.
- Receive$(t, i, m_t, b)$ expresses that message $i \in I^*$ of type $m_t \in M$ is received from business partner $b \in B$ at $t \in \Sigma$.
- Send$(t, i, m_t, b)$ expresses that message $i \in I^*$ of type $m_t \in M$ is sent to business partner $b \in B$ at $t \in \Sigma$.
- Parameter$(t, i, p, v)$ expresses that execution parameter $p \in \mathbb{P}$ of task or message $i \in T^* \cup I^*$ has parameter value $v \in \Omega$.
- Write$(t, i, p, v, d)$ expresses that task or message $i \in T^* \cup I^*$ writes value $v \in \Omega$ to data container $d \in D$ via output parameter $p \in \mathbb{P}$ at $t \in \Sigma$.
- Read$(t, i, p, v, d)$ expresses that task or message $i \in T^* \cup I^*$ reads value $v \in \Omega$ to data container $d \in D$ via input parameter $p \in \mathbb{P}$ at $t \in \Sigma$.

Based on predicate Write, we define predicate Value:

$$\text{Value}(t_0, a, o) \iff \exists t_1 \in \Sigma: t_1 < t_0 \wedge \text{Write}(t_1, \cdot, \cdot, v, a) \wedge \exists t_2: t_1 < t_2 < t_0 \wedge \text{Write}(t_2, \cdot, \cdot, o)$$

Note that we use dots $(\cdot)$ as wildcards, i.e.:

$$\text{Write}(t_1, \cdot, \cdot, o) \iff \exists x_1, x_2, x_3: \text{Write}(t_1, x_1, x_2, x_3, o)$$
4.3 Formal Description of eCRG

We now introduce a set-based formal description of the eCRG. Since the latter constitutes a graph, it consists of nodes (e.g., task nodes) and edges (e.g., sequence flow edge). Furthermore, the eCRG nodes and edges may have attachments that specify additional constraints. Thus, we can roughly define an eCRG as specified in Def. 3:

**Definition 3 (eCRG).**

An eCRG $\Psi$ is a graph $\Psi = (N, E, \bigodot)$, where
- $N$ is the set of nodes,
- $E$ is the set of edges, and
- $\bigodot$ is the set of attachments to nodes and edges.

As described in Section 3, each element of an eCRG (i.e., node, edge, or attachment) can either be element of the antecedence pattern $(A)$ or the consequence pattern $(C)$ or be a particular instance $(I)$ (e.g., a particular employee Mr. Smith, date 3rd January 2013, or role supervisor). In the cases of nodes, the antecedence pattern is partitioned into the antecedence occurrence pattern $(AO)$ and the antecedence absence pattern $(AA)$, while the consequence pattern is partitioned into the consequence occurrence pattern $(CO)$ and the consequence absence pattern $(CA)$. Before formally introducing these pattern classes in Def. 7, we first define the various elements of an eCRG (i.e., nodes, edges and attachments) in more detail.

**Nodes.** The most fundamental elements of the eCRG are nodes. According to the different compliance perspectives the set of nodes $N$ can be partitioned into nodes corresponding to tasks, receiving and sending messages, points in time, data containers, data objects, and resources as follows:

**Definition 4 (Nodes of an eCRG).**

Let $N$ be the set of nodes of an eCRG $\Psi = (N, E, \bigodot)$. Then, $N$ can be partitioned as follows:

$N := T \cup M_R \cup M_S \cup P \cup D \cup O \cup R$, where

- $T$ is the set of task nodes,
- $M_R / M_S$ is the set of receiving/sending message nodes,
- $P$ is the set of nodes representing points in time,
- $D$ is the set of data container nodes,
- $O$ is the set of data object nodes,
- $R := R_S \cup R_G \cup R_R \cup R_U$ are the sets of resource nodes for staff members, groups, roles, and organizational units.

In the following, we use $T_S \cup T_E$ instead of $T$ in order to distinguish whether edges are connected to the left and right side of task nodes. Thereby, $T_S$ is the left side, i.e., it corresponds to the start of a task, whereas $T_E$ is the right side corresponding to completion end of a task. Furthermore, we introduce the function $s : T \rightarrow T_S$ ($e : T \rightarrow T_E$) to get the start (end) of a task node, and function $t : T_S \cup T_E \rightarrow T$ for the opposite direction. Function $type : (T \rightarrow T) \cup (M \rightarrow M)$ assigns to each task and message node the associated task or message
type. \( hp: (M_R \cup M_S) \rightarrow B \) denotes the business partner that sends or receives a message. Finally, \( TMP := T \cup M_R \cup M_S \cup P \subseteq N \) is the set of a task, message, and points in time nodes, whereas \( DR := D \cup O \cup R \subseteq N \) comprises all data container, data object, and resource nodes.

We can partition the set of nodes and each of its subsets into distinct sets of particular instance nodes, antecedence nodes, and absence nodes as well as into particular instance nodes, antecedence occurrence nodes, antecedence absence nodes, consequence occurrence nodes, and consequence absence nodes. We use the exponents \( I, A, C, AO, AA, CO, C \) to indicate the corresponding subset. Further, the exponent \(-I\) denotes that all nodes are considered except particular instance ones.

**Edges.** The nodes of an eCRG can be connected through a wide range of edges indicating relations and correlations between these nodes. In particular, the set of edges can be partitioned into distinct subsets (cf. Def. 5). The latter contain edges for sequence flow, message flow, and distance in time. Other edge subsets refer to read/write data flow from/to data containers or data objects as well as to the performers of tasks and relations among data objects or resources. Finally, the two last subsets contain edges specifying correlations between conditions to data containers and the particular points in time, when these conditions are evaluated:

**Definition 5 (Edges of an eCRG).**

We consider \( E \) as the set of edges of an eCRG \( \Psi = (N, E, \Phi) \). Then, \( E \) can be partitioned as follows:

\[
E := \mathsf{sf} \cup \mathsf{tmf} \cup \mathsf{df} \cup \mathsf{wdf}\cup \mathsf{rdf}\cup \mathsf{rpm}\cup \mathsf{pr}\cup \mathsf{dr}\cup \mathsf{dce}\cup \mathsf{dco},
\]

where

- \( \mathsf{sf} \subseteq (T \cup M_R \cup M_S) \times X_T \) is the set of sequence flow edges including a time condition,
- \( \mathsf{tmf} \subseteq M_M \times M_R \) is the set of message flow edges,
- \( \mathsf{df} \subseteq (T \cup M_R \cup M_S \cup P) \times X_T \) is the set of time distance edges including a time condition,
- \( \mathsf{wdf}\subseteq (T \cup M_R \cup M_S) \times (\mathcal{P} \cup \{\}\}) \times D \) is the set of data flow edges writing to a data container,
- \( \mathsf{rdf}\subseteq D \times (T \cup M_R \cup M_S) \times (\mathcal{P} \cup \{\}) \) is the set of data flow edges reading from a data container,
- \( \mathsf{wdf}\subseteq (T \cup M_R \cup M_S) \times (\mathcal{P} \cup \{\}) \times O \) is the set of data flow edges writing a data object,
- \( \mathsf{rdf}\subseteq O \times (T \cup M_R \cup M_S) \times (\mathcal{P} \cup \{\}) \) is the set of data flow edges reading a data object,
- \( \mathsf{rpm}\subseteq T \times R \) is the set of performing relation edges denoting the performer of a task,
- \( \mathsf{pr}\subseteq R^2 \times X_R \) is the set of resource relation edges,
- \( \mathsf{dce}\subseteq (T \cup M_R \cup M_S \cup P) \times D \times X_D \) is the set of data condition edges constraining a data container at a particular point in time by a data condition,
- \( \mathsf{dco}\subseteq (T \cup M_R \cup M_S \cup P) \times D \times O \) is the set of data condition edges requiring a data container to contain a data object at a particular point in time.

Further, \( \mathsf{src} : E \rightarrow N \) (\( tgt : E \rightarrow N \)) is a function that maps each edge to its source node (target node), whereas \( \mathsf{nd} : E \rightarrow N, e \mapsto \mathsf{nd}(e) := \{ \mathsf{src}(e), tgt(e) \} \) is a function that maps each edge to the set containing its source and target node.

Edges are either part of the antecedence or the consequence pattern. Hence, we can further partition the set of edges and each of its subsets into antecedence and
consequence edges. We use exponents $A$ and $C$ to indicate when only considering the particular subset.

According to the definition of $s_j^f$, each sequence flow requires a time relation. Thereby, if the graphical representation does not provide a time relation, we just assume the time condition to be $T^2$, i.e., true for each input. Further an antecedence edge with a consequence time condition is interpreted as two edges: an antecedence edge without a time condition and a consequence edge with the respective time condition.

Attachments. Nodes and edges of an eCRG may be refined by attachments. In turn, these attachments constitute additional conditions for the respective node or edge. In particular, the set of attachments $\triangle$ can be partitioned into distinct subsets. The latter refer to different data conditions constraining data flow or parameters of tasks or messages. Other attachments express time conditions that constrain the duration of tasks or resource conditions that constrain resources:

Definition 6 (Attachments of an eCRG).
We consider $\triangle$ as the set of attachments of an eCRG $\Psi = (N, E, \triangle)$. Then, $\triangle$ can be partitioned as follows:

- $\triangle_{dcp} \subseteq (T \cup M) \times P \times C_D$ attaches a task or message node with a data condition on a particular parameter,
- $\triangle_{doc} \subseteq O \times C_D$ attaches data object nodes with a data condition,
- $\triangle_{dfc} \subseteq (wdfc \cup rdfc \cup wdf0 \cup rdf0) \times C_D$ attaches data flow edges with a data condition,
- $\triangle_{dfo} \subseteq (wdfc \cup rdfc) \times O$ attaches data flow edges from/to a data container with a data object serving as placeholder for the value of the data flow,
- $\triangle_{td} : T \times RT$ attaches task nodes with a time relation that constrains the duration of the task, and
- $\triangle_{rc} : R \times CR$ attaches resource nodes with a resource condition.

Further, $at : \triangle \rightarrow N \cup E$ is a function mapping each attachment to the node or edge it corresponds to. Attachments are either part of the antecedence or the consequence pattern. Consequently, we can partition the set of attachments and each of its subsets into antecedence and consequence attachments. Again, we use exponents $A$ and $C$ when solely referring to the respective subset.

Pattern Classes. As mentioned in Section 3, nodes, edges, and attachments can be classified as elements of the instance, the antecedence, or the consequence pattern. In the context of nodes, the antecedence and consequence pattern are further partitioned into the antecedence occurrence and the antecedence absence pattern as well as the consequence occurrence and the consequence absence pattern. Def. 7 formally introduces the pattern classes and transfers the finer classification to edges and attachments (i.e., the classification into antecedence occurrence, antecedence absence, consequence occurrence, and consequence absence patterns). For this purpose, we consider pattern classes of the nodes connected by an edge as well as the pattern class of the element an attachment corresponds to.
transformation in detail. In this context, we use relation and edges. Thus, Def. 8 introduces the variables required, before describing the resulting logic formula comprises variables derived from the eCRG elements.

After having defined a formal representation of the process environment, process logs and compliance rules, we are able to define the semantics of a particular compliance rule. For this purpose, we transform the eCRG into a first-order predicate logic formula in this subsection. Instead of nodes, edges and attachments, we replace them with variables derived from the eCRG elements. However, note that the set of variables is not isomorphic to the sets of nodes and edges. Thus, Def. 8 introduces the variables required, before describing the transformation in detail. In this context, we use relation \( \xi \) to indicate the set of values a variable may take.

**Definition 8 (Variables).**

Let \( \Psi = (N, E, \Phi) \) be an eCRG, then:

- \( \nu^1_1 \xi \xi \) corresponds for each \( x \in T \cup T_E \cup M_R \cup M_S \cup P \) to the variable for either the point in time of the start (end) \( x \) of a task node, the point of receiving (sending) a message related to message node \( x \), the time related to a point in time node \( x \), or the particular point in time \( x \) if \( x \in P \) is a particular point in time instance node.
- \( \nu^1_2 \xi \xi \) corresponds for each \( x \in T \cup T_E \cup M_R \cup M_S \cup P \) to the variable for the instance identifier of a task or message node \( x \).
- \( \nu^1_1 \xi \xi \) corresponds for each \( x \in T \) to the variable for the performing staff member of a task node \( x \).

---

**4.4 Transformation of eCRG**

On the Formal Semantics of the Extended Compliance Rule Graph 15

**Definition 7 (Classes of an eCRG).**

Let \( \Psi = (N, E, \Phi) \) be an eCRG. Then:

- \( I := N_I \) is the instance pattern,
- \( A := N^{AO} \cup N^{AA} \cup E^A \cup \Phi^A \) is the antecedence pattern,
- \( C := N^{CO} \cup N^{CA} \cup E^C \cup \Phi^C \) is the consequence pattern,
- \( E^{AO} := \{ e \in E^A \mid nd(e) \in N^{AO} \} \) is the set of antecedence occurrence edges,
- \( E^{AA} := \{ e \in E^A \mid nd(e) \cap N^{AA} = \emptyset \cap nd(e) \cap N^{AO} \neq \emptyset \} \) is the set of antecedence absence edges,
- \( E^{CO} := \{ e \in E^C \mid nd(e) \cap N^{C} = \emptyset \wedge nd(e) \cap (N^{CO} \cup N^{AO}) \} \) is the set of consequence occurrence edges,
- \( E^{CA} := \{ e \in E^C \mid nd(e) \cap N^{CA} \neq \emptyset \wedge nd(e) \cap (N^{CO} \cup N^{AO}) \} \) is the set of consequence absence edges,
- \( \Phi^{AO} := \{ a \in \Phi^A \mid at(a) \in N^{AO} \cup E^{AO} \} \) is the set of antecedence occurrence attachments,
- \( \Phi^{AA} := \{ a \in \Phi^A \mid at(a) \in N^{AA} \cup E^{AA} \} \) is the set of antecedence absence attachments,
- \( \Phi^{CO} := \{ a \in \Phi^C \mid at(a) \in N^{CO} \cup E^{CO} \cup N^{C} \} \) is the set of consequence occurrence attachments,
- \( \Phi^{CA} := \{ a \in \Phi^C \mid at(a) \in N^{CA} \cup E^{CA} \} \) is the set of consequence absence attachments,
- \( AO := N^{AO} \cup E^{AO} \cup \Phi^{AO} \) is the antecedence occurrence pattern,
- \( AA := N^{AA} \cup E^{AA} \cup \Phi^{AA} \) is the antecedence absence pattern,
- \( CO := N^{CO} \cup E^{CO} \cup \Phi^{CO} \) is the consequence occurrence pattern,
- \( CA := N^{CA} \cup E^{CA} \cup \Phi^{CA} \) is the consequence absence pattern, and
- \( \Phi := \{ I, AO, AA, CO, CA \} \) is the set of pattern classes.
transformation in Def. 10. The corresponding elements. Finally, the resulting terms are used to specify the conditions, which are derived by applying the transformation patterns.

For each pattern class \( c \in C \) (except \( I \)) and each node \( n \in N \), Def. 9 summarizes the conditions, which are derived by applying the transformation patterns \( \Gamma \) to the corresponding elements. Finally, the resulting terms are used to specify the transformation in Def. 10.

**Definition 9 (Pattern Conditions).**
Let \( \Psi = (N, E, \Phi) \) be an eCRG, let \( c \in C - \{ I \} \) be a pattern class, and let \( n \in N \) be a node. Then:

- \( \zeta^c = \bigwedge_{x \in c} \Gamma(x) \) corresponds to the conjunction of all conditions of pattern \( c \).
- \( \zeta_n^c = \bigwedge_{x \in c, \phi(x) = n} \Gamma(x) \) corresponds to the conjunction of all conditions of pattern \( c \) for node \( n \).
Table 2: Transformation Pattern for Nodes

<table>
<thead>
<tr>
<th>x</th>
<th>(I'(x))</th>
<th>(\phi(x))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\alpha \in T)</td>
<td>Start((\nu'_e(\alpha), \nu'_e, \text{type}(\alpha)) \land \text{End}(\nu'_e(\alpha), \nu'_e, \text{type}(\alpha)) \land \nu'_e(\alpha) \leq \nu'_e(\alpha))</td>
<td>(\alpha)</td>
</tr>
<tr>
<td>(\beta \in M)</td>
<td>(\text{Receive}(\nu'_e(\beta), \nu'_e, \text{type}(\beta), \text{bp}(\beta)))</td>
<td>(\beta)</td>
</tr>
<tr>
<td>(\beta \in M)</td>
<td>(\text{Send}(\nu'_e(\beta), \nu'_e, \text{type}(\beta), \text{bp}(\beta)))</td>
<td>(\beta)</td>
</tr>
<tr>
<td>(\tau \in \mathcal{P})</td>
<td>true</td>
<td>(\tau)</td>
</tr>
<tr>
<td>(\gamma \in \mathcal{R})</td>
<td>true</td>
<td>(\gamma)</td>
</tr>
<tr>
<td>(\omega \in \mathcal{D})</td>
<td>true</td>
<td>(\omega)</td>
</tr>
<tr>
<td>(\delta \in \mathcal{O})</td>
<td>true</td>
<td>(\delta)</td>
</tr>
</tbody>
</table>

Table 3: Transformation Pattern for Edges

<table>
<thead>
<tr>
<th>x</th>
<th>(I'(x))</th>
<th>(\phi(x))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e = (\alpha, \beta, \text{rel}_t) \in s^f)</td>
<td>(\nu'_e \leq \nu'_e \land \text{rel}_t(\alpha, \beta))</td>
<td>if (\alpha \in N^{AA} \cup N^{CA}) then (\alpha) else (\beta)</td>
</tr>
<tr>
<td>(e = (\alpha, \beta) \in m^f)</td>
<td>(\nu'_e = \nu'_e)</td>
<td>if (\alpha \in N^{AA} \cup N^{CA}) then (\alpha) else (\beta)</td>
</tr>
<tr>
<td>(e = (\alpha, \beta, \text{con}_t) \in t^d)</td>
<td>((\nu'_e(\alpha) \geq \nu'_e(\beta) \land \text{con}_t((\nu'_e(\beta) - \nu'_e(\alpha))))) ((\nu'_e(\alpha) \geq \nu'_e(\beta) \land \text{con}_t((\nu'_e(\beta) - \nu'_e(\alpha))))) ((\nu'_e(\alpha) &lt; \nu'_e(\beta) \land \nu'_e(\beta) &lt; \nu'_e(\alpha) \land \text{con}_t(0))))</td>
<td>if (\alpha \in N^{AA} \cup N^{CA}) then (\alpha) else (\beta)</td>
</tr>
<tr>
<td>(e = (\gamma, \eta, \text{rel}_r) \in f^r)</td>
<td>(\text{rel}_r(\gamma, \eta))</td>
<td>(\gamma)</td>
</tr>
<tr>
<td>(e = (\delta, \epsilon, \text{rel}_d) \in d^r)</td>
<td>(\text{rel}_d(\delta, \epsilon))</td>
<td>(\delta)</td>
</tr>
<tr>
<td>(e = (\alpha, \omega, \text{con}_d) \in d^c)</td>
<td>(\text{val}(\nu'_e(\alpha), \omega, \nu'_e(\alpha)) \land \text{con}_d(\nu'_e(\alpha)))</td>
<td>(\alpha)</td>
</tr>
<tr>
<td>(e = (\alpha, \omega, \delta) \in d^c)</td>
<td>(\text{val}(\nu'_e(\alpha), \omega, \nu'_e(\alpha)))</td>
<td>(\alpha)</td>
</tr>
</tbody>
</table>

Table 4: Transformation Pattern for Attachments

<table>
<thead>
<tr>
<th>x</th>
<th>(I'(x))</th>
<th>(\phi(x))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a = (\alpha, \text{p, con}_d) \in ^* \text{dep})</td>
<td>(\text{Parameter}(\nu'_e(\alpha), \nu'_e, \nu'_e) \land \text{con}_d(\nu'_e))</td>
<td>(\alpha)</td>
</tr>
<tr>
<td>(a = (\alpha, \text{con}_d) \in ^* \text{doc})</td>
<td>(\text{con}_d(\nu'_e))</td>
<td>(\phi(\alpha))</td>
</tr>
<tr>
<td>(a = (\epsilon, \text{con}_d) \in ^* \text{dfc})</td>
<td>(\text{un}_e(\alpha))</td>
<td>(\phi(\epsilon))</td>
</tr>
<tr>
<td>(a = (\epsilon, \delta) \in ^* \text{dfc})</td>
<td>(\nu'_e = \nu'_e)</td>
<td>(\phi(\epsilon))</td>
</tr>
<tr>
<td>(a = (\alpha, \text{rel}_t) \in ^* \text{td})</td>
<td>(\text{rel}_t(\nu'_e(\alpha), \nu'_e(\alpha)))</td>
<td>(\alpha)</td>
</tr>
<tr>
<td>(a = (\alpha, \text{con}_r) \in ^* \text{rc})</td>
<td>(\text{con}_r(\nu'_e))</td>
<td>(\gamma)</td>
</tr>
</tbody>
</table>
exalt corresponds to a particular consequence pattern and \(\exists\) multiple consequences in Def. 11. Hence, we formally introduce exclusive and alternative connectors as well as exclusive and alternative connectors nor with multiple consequence patterns. Note that the transformation specified in Def. 3-10 is neither able to deal with dummy variable.

An exclusive connector is satisfied if the condition related to exactly one of its nodes (and connected edges and attachments) is satisfied. In turn, the conditions related to the other nodes of the exclusive connector must be violated. Thus, to

\[\begin{align*}
\text{Definition 10 (Transformation).} \\
\text{The semantic of an eCRG } \Psi = (N, E, \Phi) \text{ is expressed by the following FOL formula } A(\Psi): \\
A(\Psi) := \forall \Theta(V_{\text{TMP}}^{\text{AO}}): \left( \left( \exists \Theta(V_{\text{RD}}^{\text{AO}}): \xi \right) \land \left( \bigwedge_{n \in \text{TMP}A} \exists \Theta(V_{n}^{\text{AA}}) : \exists \Theta(V_{n}^{\text{AO}} \cup V_{n}^{\text{RA}}) : \xi^{\text{AO}} \land \zeta^{\text{AA}} \right) \right) \\
\Rightarrow \exists \Theta(V_{\text{TMP}}^{\text{CO}}): \left( \left( \exists \Theta(V_{\text{RD}}^{\text{AO}} \cup V_{\text{RD}}^{\text{CO}}): \xi^{\text{AO}} \land \xi^{\text{CO}} \right) \right) \\
\land \left( \bigwedge_{n \in \text{TMP}C} \exists \Theta(V_{n}^{\text{CA}}): \exists \Theta(V_{n}^{\text{AO}} \cup V_{n}^{\text{CO}} \cup V_{n}^{\text{CA}}) : \xi^{\text{AO}} \land \xi^{\text{CO}} \land \zeta^{\text{CA}} \right) \right) \\
\end{align*}\]

Thereby, \(\Theta(W)\) is replaced by the comma-separated concatenation of the variables contained by a set \(W \subset V\). For example \(\exists \Theta((\nu_1, \nu_2, \nu_3))\) would be replaced by \("\exists\nu_1, \nu_2, \nu_3"\). In case of the empty set (i.e., \(W = \emptyset\)), \(\Theta(W)\) is replaced by a dummy variable.

Note that the transformation specified in Def. 3-10 is neither able to deal with exclusive and alternative connectors nor with multiple consequence patterns. Hence, we formally introduce exclusive and alternative connectors as well as multiple consequences in Def. 11.

\[\begin{align*}
\text{Definition 11 (eCRG with multiple consequences, exclusive & alternative connectors).} \\
\text{An eCRG with exclusive and alternative connectors is a tuple } \Psi^+ = (N, E, \Phi, \pi, \alpha, \omega). \text{ Thereby } \Psi^+ = (N, E, \Phi) \text{ is an eCRG and} \\
\begin{align*}
&- \pi \in 2^{T \cup MS \cup MR^{\text{up}}} \text{ is the set of exclusive connectors between task nodes, receiving message nodes, and points in time} \\
&- \alpha \in 2^{T \cup MS \cup MR^{\text{up}}} \text{ is the set of alternative connectors between task nodes, sending message nodes, and points in time and} \\
&- T, M, P \subseteq N \text{ are the distinct sets of task nodes, sending message nodes, receiving message nodes, and points in time of } \Psi^+ \text{ (cf. Def. 4). We define} \\
&- \chi \in T \text{ the set of nodes contained in the set of exclusive connectors}, \\
&- \zeta \in T \text{ the set of nodes contained in the set of alternative connectors}, \\
&- \eta \in T \text{ as the union of these two sets.} \\
\end{align*}
\end{align*}\]

Exclusive and alternative connectors either connect elements of the antecedence pattern solely or elements of the same consequence pattern. Consequently, we can partition the set of exclusive and alternative connectors as well as the related sets of nodes \((N_{\text{ex}}, N_{\text{alt}}, N_{\text{ex,alt}})\) into antecedence and consequence connectors.

We use exponents \(^{\text{ex}}\) and \(^{\text{alt}}\) to indicate the corresponding subset.

An exclusive connector is satisfied if the condition related to exactly one of its nodes (and connected edges and attachments) is satisfied. In turn, the conditions related to the other nodes of the exclusive connector must be violated. Thus, to
solve an exclusive connector we can split it into different cases, whereby each case corresponds to the selection of one node. In this context, the selected node remains unchanged, while all other nodes of the exclusive connector are negated (i.e. occurrence nodes become absence nodes and vice versa). Finally, the logic disjunction of all cases expresses the semantics of the exclusive connector. An alternative connector is satisfied, if the condition related to at least one of its nodes (and connected edges and attachments) is satisfied. In turn, the conditions of the remaining nodes are irrelevant. According to exclusive connectors, we can solve an alternative connector by splitting it into its different cases, whereby each case corresponds to the selection of one node. In this context, the selected node remains unchanged, whereas the remaining nodes of the alternative connector are removed. Again, the logic disjunction of all cases expresses the semantics of the alternative connector. In Def. 12 formally introduce selections of nodes first, which correspond to the cases mentioned above.

**Definition 12 (Selections).**

Let $\Psi^* = (N, E, \Phi, e, \bar{e}, \text{ex, alt})$ be an eCRG with multiple consequences, exclusive and alternative connectors. Then

- $\tau \subseteq N^A$ is an antecedence selection choosing one node out of each exclusive and alternative antecedence connector; i.e., $\forall X \in \text{ex}^A \cup \text{alt}^A : |\tau \cap X| = 1$,
- $\tau^{-1} = N^A_{\text{ex, alt}} - \tau$ is the complement of $\tau$,
- $\upsilon \subseteq N^{C_i}$ is a consequence selection choosing one node out of each exclusive and alternative consequence connector of $C_i$; i.e., $\forall X \in \text{ex}^{C_i} \cup \text{alt}^{C_i} : |\upsilon \cap X| = 1$,
- $\upsilon^{-1} = N^{C_i}_{\text{ex, alt}} - \upsilon$ is the complement of $\upsilon$,
- $\Pi^A = \{ \tau \subseteq N^A_{\text{ex, alt}} \cap (N^{AO} \cup N^{AA}) | \forall X \in \text{ex}^A \cup \text{alt}^A : |\tau \cap X| = 1 \}$ is the set of all antecedence selections, and
- $R^{C_i} = \{ \upsilon \subseteq N^{C_i}_{\text{ex, alt}} (N^{CO} \cup N^{CA}) | \forall X \in \text{ex}^{C_i} \cup \text{alt}^{C_i} : |\upsilon \cap X| = 1 \}$ is the set of all consequence selections of consequence pattern $C_i$.

As aforementioned, each selection configures the pattern class of nodes; i.e. selected nodes remain in their class, while the non-selected ones change their pattern class (exclusive connector) or are removed (alternative connector). Def. 13 specifies these node configurations in detail.

**Definition 13 (Configuration of Nodes).**

Let $\Psi^* = (N, E, \Phi, e, \bar{e}, \text{ex, alt})$ be an eCRG with multiple consequences, exclusive and alternative connectors, and let $\tau \in \Pi^A$ be an antecedence selection and $\upsilon \in R^{C_i}$ be a consequence selection. Then

- $N^{AO, \tau} := (N^{AO} - N^A_{\text{ex, alt}}) \cup (\tau \cap N^{AO}) \cup (\tau^{-1} \cap N^{AO})$ corresponds to the configuration of antecedence occurrence nodes $N^{AO}$ based on $\tau$,
- $N^{AA, \tau} := (N^{AA} - N^A_{\text{ex, alt}}) \cup (\tau \cap N^{AA}) \cup (\tau^{-1} \cap N^{AA})$ corresponds to the configuration of antecedence absence nodes $N^{AA}$ based on $\tau$,
- $N^{CO, \upsilon} := (N^{CO} - N^{C_i}_{\text{ex, alt}}) \cup (\upsilon \cap N^{CO}) \cup (\upsilon^{-1} \cap N^{CA})$ corresponds to the configuration of antecedence occurrence nodes $N^{CO}$ based on $\tau$,
- $N^{CA, \upsilon} := (N^{CA} - N^{C_i}_{\text{ex, alt}}) \cup (\upsilon \cap N^{CA}) \cup (\upsilon^{-1} \cap N^{CO})$ corresponds to the configuration of antecedence absence nodes $N^{CA}$ based on $\tau$. 

On the Formal Semantics of the Extended Compliance Rule Graph 19
Based on Def. 14, we are able to enrich the transformation from Def. 10 with the support for exclusive and alternative connectors as well as multiple consequences in Def. 15.
For example, Def. 10 and Def. 15 translate eCRG BPCPs. Fig. 12 provides eCRGs that model the 7 resource BPCPs and the 5 time BPCPs focus on the resource perspective, and 5 BPCPs focus on the time perspective. They include all property specification patterns from [54]. Out of the 27 BPCPs, 15 BPCPs focus on the control flow (i.e., process) perspective, 7 BPCPs focus on the resource perspective, and 5 BPCPs focus on the time perspective. In turn, the data perspective is implicitly supported by each BCP, since BPCPs do not distinguish between tasks and data conditions. Fig. 11 shows how to model the 15 control flow BPCPs with the eCRG language. In turn, Fig. 12 provides eCRGs that model the 7 resource BPCPs and the 5 time BPCPs.

5 Pattern-based Evaluation

This section outlines a pattern-based evaluation of the eCRG language. In particular, we model the compliance patterns introduced in [9, 11, 29] with the eCRG language. These patterns result from literature and case studies, and thus constitute a suitable empirical basis for evaluating the appropriateness of our approach.

5.1 Business Process Control Pattern and Property Specification Pattern

27 business process control patterns (BPCPs) for modeling compliance rules are introduced in [11]. They include all property specification patterns from [54].
Fig. 11: control flow BPCPs

Fig. 12: Resource and time BPCPs
Note that most BPCPs can easily be modeled using the eCRG language. However, from our subjective point of view the modeling of the XLeadsTo BPCP, the Universal BPCP, the Exists Max/Min BPCP, and the Exists Every k BPCP was a bit more complicated. In turn, Fig. 12 provides an eCRG for the special case of the Multi-Segregated BPCP, which requires the numbers of performers and tasks to be equal. Other cases of the Multi-Segregated BPCP can not be modeled using only one consequence pattern; e.g., in case that the Multi-Segregated BPCP requires 4 tasks to be executed by 3 different performers (i.e., one performer must execute two tasks), \( \binom{4}{3} = 6 \) consequence patterns are required. Hence the Multi-Segregated BPCP is the only BPCP that cannot always be appropriately modeled using the eCRG language.

5.2 Compliance Rule Pattern

55 control flow compliance rule patterns (CRPs) are introduced in [9]. Further, one example of the data perspective and another one of the resource perspective are presented. The 55 control flow CRPs are partitioned into 15 categories. Note that 7 categories and the respective CRPs fully coincide with the aforementioned BPCPs (cf. Table 5). The remaining 8 categories and respective CRPs are modeled with the eCRG language in Figs. 13-16. Fig. 17 contains both examples illustrating the data and resource perspectives.

<table>
<thead>
<tr>
<th>Compliance rule pattern</th>
<th>Business process control pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existence - Event universality</td>
<td>Exists</td>
</tr>
<tr>
<td>Existence - Event absence</td>
<td>Absent</td>
</tr>
<tr>
<td>Exclusive</td>
<td>Exclusive</td>
</tr>
<tr>
<td>Mutual Exclusive</td>
<td>MutexChoice</td>
</tr>
<tr>
<td>Prequisite</td>
<td>CoAbsent</td>
</tr>
<tr>
<td>Inclusive</td>
<td>CoExists</td>
</tr>
<tr>
<td>Substitute</td>
<td>LeadsTo-Else</td>
</tr>
<tr>
<td>CoRequisite</td>
<td>CoRequisite</td>
</tr>
</tbody>
</table>
Bounded Existence
Exactly k times

Between
After
Simultaneously or after

Between
Simultaneously or before

Between
Directly after

Between
Simultaneously

Between
At least one other activity

Fig. 13: Bounded existence CRPs and between CRPs

Fig. 14: Chain precedence CRPs and chain response CRPs
Fig. 15: Bounded sequence CRPs and parallel CRPs
Fig. 16: Precedence CRPs and response CRPs

Fig. 17: Data flow CRPs and resource CRPs
Note that most CRPs can easily be modeled using the eCRG language. However, from our subjective point of view a thinking outside the box was required when we modeled the *Bounded Existence - Exactly k times in a row* CRP, the *Bounded Existence - At most k times in a row* CRP, the *Between - Simultaneously* CRP, and all *Bounded Sequence* CRPs - especially in case of the *Bounded Sequence - One to one coexistence* CRP.

5.3 Time Pattern

In [28, 29], a set of 10 *time patterns* (TPs) is introduced and formally specified. In Fig. 18, the eCRG language is applied to model these TPs.

Note that most TPs can be easily modeled using the eCRG language. However, from our subjective point of view the modeling of *TP6 Time-based Restrictions* and the modeling of *TP8 Time-dependent Variability* were a bit more complicated.

Fig. 19 summarizes the results of the pattern-based evaluation. Overall, we were able to model 26 out of the 27 business process control patterns (BPCPs) [11], including 5 time BPCPs and 7 resource BPCPs as well. Only the multi-segregation pattern cannot be properly modeled using the eCRG language in any case. Further, the eCRG language supports the 15 categories of compliance rule patterns.
and respective rules as well as the data flow and the resource example from [9]. Finally, the eCRG language covers well-known time-patterns [29] as well.

![Fig. 19: Support of compliance patterns](image)

6 Summary and Outlook

While compliance rule modeling has been introduced by a plethora of approaches, the data, time, resource and interaction perspectives of compliance rules have not been sufficiently addressed yet [12, 8–11]. This report has introduced the extended Compliance Rule Graph (eCRG) language, which is based on the compliance rule graph (CRG) language [15, 24]. As opposed to existing visual compliance rule notations, the eCRG supports multiple perspectives; i.e., beyond the control flow perspective, data, time, resources, and interactions with business partners are considered. To provide tool support for both the modeling and verification of compliance rules, we have formalized the syntax and semantics of the eCRG. Finally, we have conducted a pattern-based evaluation to prove the proper expressiveness of the eCRG language.

In future work, we will conduct experiments to evaluate the usability and scalability of the eCRG. Furthermore, we will develop techniques for verifying compliance of business processes and process choreographies with such rules.

References