Diagnostic Information in Compliance Checking

Elham Ramezani, Dirk Fahland, and Wil M. P. van der Aalst

Eindhoven University of Technology, The Netherlands
{e.ramezani,d.fahland,W.M.P.v.d.Aalst}@tue.nl

Abstract. Compliance checking is gaining importance as today's organizations need to show that operational processes are executed in a controlled manner while satisfying predefined (legal) requirements. Deviations may be costly and expose the organization to severe risks. Compliance checking is of growing importance for the business process management and auditing communities. This paper presents a *comprehensive compliance checking approach based on Petri-net fragments and alignments*. 55 control flow oriented compliance rules, distributed over 15 categories. We formalize them in terms of Petri-net fragments describing the compliant behavior. To check compliance with respect to a rule, the event log describing the observed behavior is aligned with the corresponding fragment. The approach is *flexible* (easy to add new patterns), *robust* (the selected alignment between log and fragment is guaranteed to be optimal), and allows for both a *quantification of compliance* and *intuitive diagnostics* explaining deviations at the level of alignments. The approach can also handle resource-based and data-based compliance rules and is supported by *ProM* plug-ins.

Keywords: compliance checking, process mining, conformance checking, Petri-nets

1 Introduction

Business processes need to comply with regulations and laws set by both internal and external stakeholders. Failing to comply may be costly, therefore, organizations need to continuously check whether business processes are executed within the boundaries set by managers, governments, and other stakeholders. Deviations of the observed behavior from the specified behavior may point to fraud, malpractice, risks, and inefficiencies. Three types of compliance-related activities can be identified [23, 29, 19, 28]:

- *compliance elicitation*: determine the constraints that need to be satisfied (i.e., rules defining the boundaries of compliant behavior),
- *compliance formalization*: formulate precisely the compliance requirements derived from laws and regulations in compliance elicitation,
- *compliance implementation*: implement and configure information systems such that they fulfil compliance requirements,

- *compliance checking*: investigate whether the constraints will be met (forward compliance checking) or have been met (backward compliance checking), and
- *compliance improvement*: modify the processes and systems based on the diagnostic information in order to improve compliance.

There are two basic types of conformance checking: (1) *forward compliance checking* aims to design and implement processes where conformant behavior is enforced and (2) *backward compliance checking* aims to detect and localize non-conformant behavior. This paper focuses on backward compliance checking based on event data.

Compliance checking is gaining importance because of the availability of event data and new legislations. Major corporate and accounting scandals including those affecting Enron, Tyco, Adelphia, Peregrine and WorldCom have fueled the interest in more rigorous auditing practices. Legislation, such as the Sarbanes-Oxley (SOX) Act of 2002 and the Basel II Accord of 2004, was enacted as a reaction to such scandals. At the same time, new technologies are providing opportunities to systematically observe processes at a detailed level. Today, event data is everywhere – in every system and in every organization – and will continue to grow exponentially.

Process mining techniques [1] offer a means to more rigorously check compliance and ascertain the validity and reliability of information about an organization's core processes. The core challenge is to compare the prescribed behavior (e.g., a process model or set of rules) to observed behavior (e.g., audit trails, workflow logs, transaction logs, message logs, and databases). For example, in [3] it is shown how constraints expressed in terms of Linear Temporal Logic (LTL) can be checked with respect to an event log. In [25] both LTL-based and SCIFF-based (i.e., abductive logic programming) approaches are used to check compliance with respect to a declarative process model and an event log. Dozens of approaches have been proposed to check conformance given a Petri-net and an event log [2, 8, 6, 7, 11, 13, 20, 26, 27, 30, 37]. Approaches such as in [30] replay the event log on the model while counting "missing" and "remaining" tokens. The former indicates observed, but disallowed behavior, and the latter indicate non-observed, but required behavior. State-of-the-art techniques in conformance checking retrieve this information by computing *optimal alignments* [2, 8] between traces in the event log and "best fitting" paths in the model.

Existing approaches to backwards compliance checking have two main problems. First of all, the *elicitation of compliance rules is not supported well.* End users need to map compliance rules onto expressions in temporal logic or encode the rules into a Petri-net-like process model. Second, existing checking techniques can discover violations but *do not provide useful diagnostics.* While forward compliance checking techniques [10, 18] employ pattern matching to highlight compliance violations in a model, such techniques are not applicable in backwards checking where not a model, but a log is given. Here, LTL-based checkers will classify a trace as non-compliant without providing detailed diagnostics and discard the remainder of the trace when the first deviation is detected.

To address these limitations we provide a *comprehensive collection of control flow related compliance rules*. We identify 55 rules distributed over 15 categories. These compliance rules are formalized in terms of Petri-net patterns. We apply the alignment technique developed in [2, 8] to analyze if the process execution (log) has been compliant with the compliance rules (Petri-net patterns). If the observed behavior is consistent with the compliance rule, then the optimal alignment shows that all moves of the log can be mimicked by the corresponding Petri-net pattern and vice versa. If this is not possible because the compliance rule is violated, then the alignment shows the root cause of the deviation. This way, we are able to show *detailed diagnostics without false negatives* (non-conformant behavior remains undetected) and *false positives* (conformant behavior is classified as non-conforming because of an incorrect alignment of log and model/rule). The approach is *extendible*, i.e., to add a new type of rule, one just needs to add the Petri-net pattern to our repository. Moreover, as shown in this paper, our approach can be used to support resource-based and data-based compliance rules.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 explains the notion of alignment to relate observed and modeled behavior. Our compliance rule framework is introduced in Section 4. The primary focus of this report is on control flow compliance rules; therefore we give a list of all compliance rule categories and their containing rules in Section 5. The compliance rules' formalization in form of Petri-net fragments is presented for each compliance rule in section 6. In section 7 we discuss that the approach also supports the other perspectives (e.g. resources and data). In Section 8 the implementation of the approach in ProM is showcased. Section 9 concludes the report.

2 Related Work

The importance of compliance management has been pointed out by various authors [5]. In [29] a life cycle is introduced to structure the process of compliance management. A comparative analysis over different compliance management solution frameworks is provided in [23].

Compliance management has gained wide interest from the Business Process Management (BPM) community. Compliance checking approaches can be mapped onto two main categories [22]:

- *Forward* compliance checking aims at ensuring compliant process executions. Processes can be constructed to be compliant [31] or verified whether they are compliant [24]. Alternatively, compliance requirements can be transformed into monitoring rules [12] or model annotations which then are used to enforce compliant process executions [17, 38].
- *Backward* compliance checking evaluates in hindsight whether process executions did comply to all compliance rules or when and where a particular rule was violated. A variety of conformance checking techniques have been proposed to quantify conformance and detect deviations based on an event

log and process model (e.g., a Petri-net) [2, 8, 6, 7, 11, 13, 20, 26, 27, 30, 37]. Also approaches based on temporal logic [3, 25] have been proposed to check compliance

In this report, we focus on backward compliance checking and assume an event log to be present. Compared to existing approaches we provide a comprehensive collection of compliance rules. Moreover, we focus on providing diagnostic information.

3 Conformance Checking Based on Alignments

As will be shown in this paper, we provide a large repository of Petri-net patterns modeling typical compliance rules. These rules can be instantiated for a particular process, i.e., the abstract activities in the pattern are replaced by concrete activities also recorded in the event log. The log *complies* to the rule if each log trace is described by the Petri-net pattern. In case a trace is not described, we want to locate *where* the trace deviates from the pattern. This section recalls basic notions and a recent technique [2, 8] for finding deviations between log traces and formal specification (a Petri-net).

An *event log* is a *multiset* of *traces*. Each trace describes the life-cycle of a particular *case* (i.e., a *process instance*) as a sequence of *events*. An event often refers to the *activity* executed. However, event logs may store additional information about events. For example, many process mining techniques use extra information such as the *resource* (i.e., person or device) executing or initiating the activity, the *timestamp* of the event, or *data elements* recorded with the event (e.g., the size of an order).

From a formal point of view a trace σ_L is a sequence over an alphabet Σ_L, i.e., $\sigma_L \in {\Sigma_L}^*$. An event log L is a multiset of traces, i.e., $L \in \mathbb{B}({\Sigma_L}^*)$. The alphabet Σ_L is typically the set of activity names. However, when including additional perspectives, the alphabet may be extended to also contain information about data and resources. For example, $(\mathit{prepare\ decision}, \mathit{start}, \mathit{John}, \mathit{gold}, \mathit{50\ Euro}) \in \Sigma_L$ may refer to an event describing the start of activity "prepare decision" by John for a gold customer claiming 50 euro. The choice of Σ_L depends on the compliance rule that needs to be checked, e.g., for most control flow related rules it is sufficient to record the activity name.

A Petri-net pattern is essentially a specification prescribing compliant traces in a concise way. Technically, a *specification* $S \subseteq {\Sigma_S}^*$ is a finite set of traces over an alphabet Σ_S together with a mapping $\ell : \Sigma_S \to 2^{\Sigma_L} \cup \{\tau\}$ that relates each specification event in Σ_S to a set of log events in Σ_L or to τ. In this report, S is the set of firing sequences of a Petri-net and ℓ is the function that labels each transition with an activity name. S and ℓ can be described by other formalisms as well.

We use the alignment approach described in [2, 8] to relate traces in the log (i.e., observed behavior) to traces of the specification (i.e., prescribed behavior). An optimal alignment of σ_L to S, roughly speaking, is a trace σ_S that is possible according to S and that is *as similar to* σ_L *as possible*. By comparing σ_L and

σ_S, a business analyst gains an understanding on what has been done wrong in σ_L and what instead should have been done (to behave as shown in σ_S).

A given trace $\sigma_L \in L$ will be related to a trace $\Sigma_S \in S$ by pairing events in σ_L to events of Σ_S. Formally, a *move* (of σ_L and S) is a pair $(x, y) \in (\Sigma_L \cup \{\gg\}) \times (\Sigma_S \cup \{\gg\}) \setminus \{(\gg, \gg)\}$. For $x \in \Sigma_L, y \in \Sigma_S$, we call $(x, \gg)$ a *move on log*, $(\gg, y)$ a *move on specification* S, and if $x \in \ell(y)$, then (x, y) is a *synchronous move*.

An *alignment* of a trace $\sigma_L \in {\Sigma_L}^*$ to S is a sequence $\gamma = \langle (x_1, y_1) \dots (x_n, y_n) \rangle$ of moves (of σ_L and S) such that the projection $x_1 \dots x_n$ to Σ_L is the original trace σ_L, i.e., $\langle x_1 \dots x_n \rangle|_{\Sigma_L} = \sigma_L$, and the projection $\langle y_1 \dots y_n \rangle|_{\Sigma_S} = \sigma_S \in S$ is described by the specification.

For example, for a specification $S = \{\langle a, b, c, d \rangle, \langle a, c, b, d \rangle\}$ with $\ell(x) = \{x\}$ the trace $\sigma_L = \langle a, c, c, d \rangle$ has (among others), the following two alignments with events of σ_L shown at the top and events of S shown at the bottom: $\gamma_1 = \frac{a|c|\ c\ |\gg|d}{a|c|\gg|\ b\ |d}$ and $\gamma_2 = \frac{a|\gg|\gg|\ c\ |\ c\ |d}{a|\ c\ |\ b\ |\gg|\gg|d}$.

Both alignments yield the same specified trace $\sigma_S = \langle a, c, b, d \rangle \in S$. However, γ_1 is preferable over γ_2 as it maximizes the number of synchronous moves. The conformance checking problem in this setting is to find for a given trace σ_L and specification S an *optimal* alignment γ of σ_L to S s.t. no other alignment has fewer non-synchronous moves (move on log only or move on specification only). The technique of [2, 8] finds such an optimal alignment using a cost-based approach: a cost-function κ assigns each move (x, y) a cost $\kappa(x, y)$ such that a synchronous move has cost 0 and all other types of moves have cost greater than 0. Then an A*-based search on the space of (all prefixes of) all alignments of σ_L to S is guaranteed to return an optimal alignment for σ_L and S.

In such an optimal alignment, a move on log $(x, \gg)$ indicates that the trace σ_L had an event x that was not supposed to happen according to the specification S whereas a move on specification $(\gg, y)$ indicates that σ_L was missing an event $\ell(y)$ that was expected according to S. As the alignment preserves the position relative to the trace σ_L, we can locate the exact position where σ_L had an event too much or missed an event compared to S.

In the remainder of this report, we show how to leverage this approach to compliance checking. The optimal alignments between log and specification provide excellent diagnostic information and can be used to robustly quantify conformance. Thereby, the specification S will formally capture all traces that comply to a given compliance constraint. The alignment of a log trace σ_L to S can then clearly show where and how often σ_L deviates from the constraint.

In the next three sections, we provide a framework for compliance rules together with an extensive list of control flow rules in 15 different categories. Each rule has a comprehensive description and is formalized as a Petri-net pattern. In Sect. 7 we generalize this approach to data- and resource-related rules.

4 Compliance Rule Framework

A compliance rule prescribes how an internal or cross-organizational business process has to be designed or executed. It originates in explicitly stated regulations and can refer to the individual perspectives of a business process (control flow, data flow, organizational aspects) or a combination of several perspectives. We reviewed existing literature on compliance [4, 15, 9, 14, 21, 35, 19, 32, 34], collected the rules described in these papers, and categorized them. We found that a single rule usually is not concerned with only one perspective of a process, but with several perspectives. Based on this observation, we identified six orthogonal dimensions of compliance rules, into which each of the rules could be categorized. For example, the rule "After a claim of more than 3000 EUR has been filed, two different employees need to check the validity of the claim independently." is composed of 3 basic rules that refer to (1) *control flow* ("After a claim has been filed, validity must be checked."), (2) *data flow* ("A claim over 3000 EUR requires two validity checks."), and (3) the *organization* ("Multiple validity checks are carried out by different employees.").

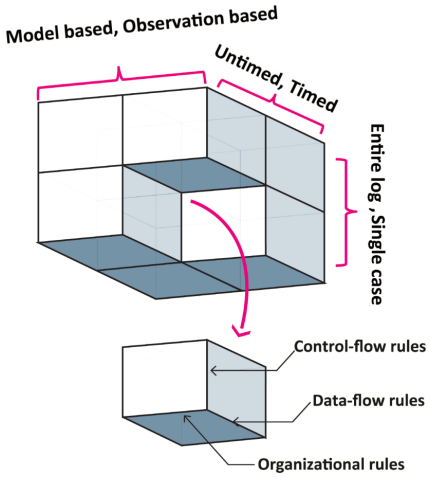

Fig. 1. Compliance Rule Framework

Furthermore, a compliance rule can (4) impose *time-related* constraints (e.g., "Within 6 months the claim must be decided.") or can be *untimed*,(5) prescribe properties of a *single case* or of *multiple* cases (e.g., "20% of all claims require a detailed check."), and (6) prescribe properties of the *process design* (e.g., "The claim process must have a time-out event handler.") or properties of the process executions, which can be *observed* (i.e., recorded in an event log).

These six basic dimensions of compliance rules are orthogonal and give rise to the framework shown in Fig. 1. In this report, we present compliance rules for control flow, data flow as well as organizational aspects, where we focus on untimed, observation-based properties of individual cases. Next section, section 5, presents an overview of control flow compliance rule categories and section 6

presents all compliance rules and their formalization in form of Petri-net patterns. In section 7, we explain data flow rules, organizational rules, and their combination with control flow rules.

5 An Extensive Collection of Control Flow Compliance Rules

Eliciting and *formalizing* compliance rules for a business process comprise determining the laws and regulations that are relevant for this process and formulating these compliance rules in an unambiguous, yet understandable manner [29]. Typically, this involves expressing a given informal requirement in a formal notation: a task an end user may not be capable of. To support elicitation, we provide end users with an *extensive library* of comprehensive compliance rules. Each rule has an informal, precise description and is accompanied by a mathematical formalization. The end user just has to pick the rule(s) that describe the given compliance requirement best; the accompanying formalization is then used for compliance checking.

We collected from literature [4, 15, 9, 14, 21, 35, 19, 32, 34] 55 compliance rules that concern the control flow perspective of a process, and classified them further into 15 categories; see Tab. 1. Each category includes several compliance rules. For example, the *Existence* category defines two rules in total, e.g., "In each process execution, task A should be executed" and "In each process execution, Task A should not be executed." Each rule is parameterized over tasks (e.g., Task A) or numeric parameters (e.g., governing bounds for repetitions etc.).

To formalize these rules we need to use a concrete formalism. Some compliance rules prescribe behaviors that are easier to express in terms of logical formulas (each A is followed by a B), and some rules prescribe behaviors that are easier to express in a more operational model (A, B, and C happen twice directly in sequence with no other event in between). Our literature survey found both kinds of rules to be relevant, temporal logics (e.g., LTL) against operational models (e.g., Petri-nets). Because of tool support for conformance checking [8], we decided to formalize rules as *parameterized Petri-net patterns*. Although being an unusual choice, we could formalize operational rules as well as declarative rules in a systematic and understandable way. The complete collection of compliance rules and their Petri-net patterns is described in the next section.

6 Petri-Net Patterns for Control Flow Compliance Rules

All the Petri-net patterns discussed in this section follow some systematics that will be explained throughout the section. Moreover there are some basic principles, we would like to present it at the beginning of this section:

- Each pattern has a dedicated place *Initial* and a place *Final*.

Table 1. Categorization of the 55 Control Flow Compliance Rules

Category (Rules)	**Description**
Existence (2)	Limits the occurrence or absence of a given event A within a scope. [4],[15],[9], [14],[21],[35],[32]
Bounded Existence (6)	Limits the number of times a given event A must or must not occur within a scope. [15],[14]
Bounded Sequence (5)	Limits the number of times a given sequence of events must or must not occur within a scope. [15],[14]
Parallel (1)	A specific set of events should occur in parallel within a scope. [32]
Precedence (10)	Limits the occurrence of a given event A in precedence over a given event B. [15],[32],[14],[35],[9],[19],[21],[4],[32]
Chain Precedence (4)	Limits the occurrence of a sequence of events $A_1, \ldots, A_n$ in precedence over a sequence of events $B_1, \ldots, B_n$. [15],[14],[21]
Response (10)	Limits the occurrence of a given event B in response to a given event A. [32],[14],[21],[15],[36],[9],[19]
Chain Response (4)	Limits the occurrence of a sequence of events $B_1, \ldots, B_n$ in response to a sequence of events $A_1, \ldots, A_n$. [15]
Between (7)	Limits the occurrence of a given event B between a sequence of events A and C. [14]
Exclusive (1)	Presence of a given event A mandates the absence of an event B. [15]
Mutual Exclusive (1)	Either a given event A or event B must exist but not none of them or both. [15],[33]
Inclusive (1)	Presence of a given event A mandates that event B is also present. [15]
Prerequisite (1)	Absence of a given event A mandates that event B is also absent. [15]
Substitute (1)	A given event B substitutes the absence of event A. [15]
Corequisite (1)	Either given events A and B should exist together or be absent together. [15]

- A token in the final place defines the final marking of the pattern. When a pattern reaches its final marking, the pattern is properly completed (i.e., all other places of the net is empty).
- Σ_L denotes the set of activity names for control flow compliance rules. Depending on the choice of compliance rules, it may include elements describing start and completion of activities[1].
- Occurrences of event(s) specified in the compliance rule are mimicked by transitions in the pattern having the same label as the events' name. Suppose a compliance rule restricts the occurrences of three specific events A, B, and C; hence the events A, B, and C are expressed as A-labeled, B-labeled, and C-labeled transitions in the pattern.
- Occurrences of any other events than the event(s) specified in the compliance rule are mimicked by the Ω-labeled transition. This way, the patterns

[1] When including additional perspectives (e.g., data, resource) the alphabet may be extended to also contain information about data and resources.

abstract from all other trace events that are not described in the compliance rule. Suppose a compliance rule restricts the occurrences of three specific events A, B, and C; hence $\Sigma_L = \Omega \cup \{A, B, C\}$ and $\Omega \cap \{A, B, C\} = \emptyset$.
- In some patterns we need to exclude the occurrence of one specific event from the events may occur in a marking, therefore we subtract that specific element from Σ_L. Suppose we need to exclude occurrence of an event A at a marking; this is shown as $\Sigma_L \setminus \{A\}$, specifying that any event but A may occur at that marking.
- Typically occurrence of activities, e.g., an activity A is represented as an atomic event A in the log. In some patterns, the respective compliance rules require to model the start and completion of activities in the Petri-net patterns. Any activity, e.g., an activity A can also be represented by two events $A-start$ (A_{st}) and $A-complete$ (A_{cmp}) indicating the start and completion of an ongoing activity. Therefore all Petri-net patterns of our collection rules come in these two flavors and can be picked based on the setting.
- A trace σ *complies* to a (rule of a) pattern if after executing σ, final transition F is enabled, and its occurrence leads to the *final marking*.
- F-labeled and τ-labeled transitions are silent transitions. In finding an optimal alignment between a trace and a Petri-net pattern, the 'alignment based technique' of Section 3, assigns the cost of zero for deviation of these two transitions.
- Arcs in patterns may have weight, the arc weight specifies how many tokens are consumed or produced as a result of firing a transition. If the arc weight is greater than one, the respective arc is annotated with the weight (i.e., a natural number); otherwise, it is assumed to be one.
- In some patterns a *reset arc* connects a place with a final transition. This arc ensures that when F fires all tokens are consumed from the respective place (even if it contains no token). We use *reset arcs* to consume all tokens from the net, thereby guaranteeing that after firing F, no transition is enabled anymore and the net is empty.
- In some patterns we connect a place to a final transition F with an *inhibitor arc*. An *inhibitor arc* ensures that F can only fire if the place at the other end of the *inhibitor arc* is empty.

6.1 Existence Category

This category contains compliance rules which limit the occurrence or absence of a given event A within a scope[2].

Event Universality .
Description: A given event A must occur within a specific scope. The compliance rule is violated if event A does not occur within the specified scope (e.g., a process instance). Figure 2 shows the Petri-net pattern that formalizes this rule.

[2] The scope can refer to a process instance (one specific case), a group of process instances or a time line.

At initial marking (i.e., a token in place *Initial*) any event may occur, but the pattern cannot terminate at this marking because the condition of the rule is not satisfied yet. If A occurs, place P is marked and any event may occur. In this situation the pattern may terminate. The transition F models that the end of the trace has been reached i.e., it occurs *after* all events of the trace occurred.

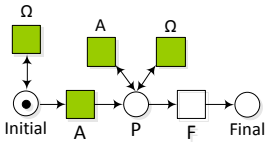

Fig. 2. 'Event Universality' Compliance rule

Event Absence .
Description: A given event A must not occur within a specific scope. The rule is violated if event A occurs within the specified scope. Figure 3 shows the Petri-net pattern that formalizes this rule.
The pattern specifies that any event but A may occur. Therefore by construction the Ω-labeled transition is always enabled. If an A occurs a deviation is captured. The transition F models that the end of the trace has been reached.

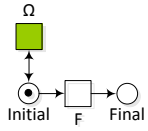

Fig. 3. 'Event Absence' Compliance rule

6.2 Bounded Existence Category

This category includes compliance rules that limit the number of times a given event A must or must not occur within a scope.

Bounded Existence of an Event. Exactly *k* times .
Description: A given event A must occur exactly k times within a scope. The rule is violated if A occurs less than or more than k times. Figure 4 shows the Petri-net pattern that formalizes this rule for the case $k = 2$.
The preplace P_k of A is initially marked with k tokens. The k tokens in place P_k assure that event A can occur at most k times, as each occurrence of A decrements the number of tokens in P_k and increments the number of tokens in place *Count*. Place *Count* counts the occurrences of A. After k times occurrences of A, P_k is empty and *Count* contains k tokens. In this situation transition A is not enabled anymore. The pattern can terminate only if there are k tokens in place *Count* implying that the condition of the rule is satisfied. Firing transition F also removes the token from place *Initial* thereby ensuring the proper completion of the pattern.

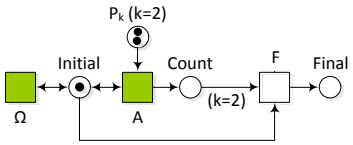

Fig. 4. 'Bounded Existence of an Event. Exactly k times' Compliance rule

Bounded Existence of an Event. At least k times .
Description: A given event A must occur at least k times within a scope. If A occurs less than k times, the rule is violated. Figure 5 shows the Petri-net pattern that formalizes this rule for the case $k = 2$.
The basic structure of the pattern in Figure 5 is similar to the pattern described in Figure 4. Such that the Ω-labeled transition is always enabled and the k tokens in place P_k limits the occurrence of the very left A-labeled transition to k times. After k occurrences of A the condition of the rule is satisfied. The pattern specifies that A must occur at least k times, therefore after k occurrences of A, further occurrences of A by consuming a token from place P are possible.
The pattern can terminate by firing transition F. Firing transition F also removes the token from place *Initial*, thereby ensuring the proper completion of the pattern.

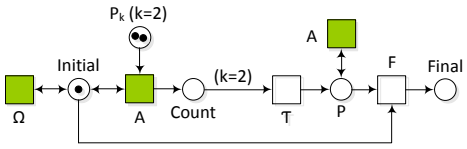

Fig. 5. 'Bounded Existence of an Event. At least k times' Compliance rule

Bounded Existence of an Event. At most k times .
Description: A given event A must occur at most k times within a scope. If A occurs more than k times, the rule is violated. Figure 6 shows the Petri-net pattern that formalizes this rule for the case ($k = 2$).
The basic structure of this pattern is similar to patterns showed in Figure 4. However in contrast with the pattern described in Figure 4, this rule allows for less than k occurrences of A. The preplace P_k of A is initially marked with k tokens which limits the occurrences of A to at most k times. After k occurrences of A, the A-labeled transition is not enabled anymore. The transition F models that the end of the trace has been reached and it is enabled if there is any number of tokens in place P_k (k, less than k or zero tokens). The *reset arc* (represented as a two arrow headed line) from place P_k to F removes the possible remaining tokens in place P_k, thereby ensuring the proper completion of the pattern.

Bounded Existence of an Event. Exactly k times in a row .
Description: A given event A must occur exactly k times in a row (directly one after the other) within a scope. The rule is violated if the sequence $\langle \underbrace{A, \ldots, A}_{k} \rangle$

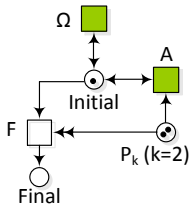

Fig. 6. 'Bounded Existence of an Event. At most k times' Compliance rule

does not occur within the specified scope. Figure 7 shows the Petri-net pattern that formalizes this rule for the case $k = 2$.
Being in the initial marking, any event may occur. As soon as the first A occurs in the log, the 'alignment based technique' of Section 3, first fires the very left τ-labeled transition and then allows k occurrences of A. After the first occurrence of A no Ω-labeled transition is enabled anymore until A occurs k times in a row. The k tokens in preplace P_k of A limits the occurrences of A to k times (similar to the structure described in Figure 4). The place *Count* counts the number of occurrences of A. Only by consuming k tokens from place *Count* the very right τ-labeled transition can fire, implying k occurrences of A. In this situation the condition of the rule is satisfied and any non-A event may occur (captured by the Ω-labeled transition) by consuming a token from place P_2. The transition F models that the end of the trace has been reached. Firing transition F removes the remaining token in place P_1, thereby ensuring the proper completion of the pattern.

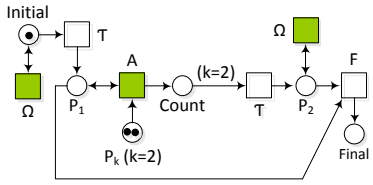

Fig. 7. 'Bounded Existence of an Event. Exactly k times in a row' Compliance rule

Bounded Existence of an Event. At least k times in a row .
Description: A given event A must occur at least k times in a row (directly one after the other) within a scope. This rule is violated if A occurs less than k times in a row within the specified scope. Figure 8 shows the Petri-net pattern that formalizes this rule for the case $k = 2$.
Being in the initial marking, any event may occur. As soon as the first event A occurs, the token from the place *Initial* is consumed and place P_1 is marked with a token (from now on place *Count* counts occurrences of A). At this marking, we distinguish two scenarios: *Scenario1* is that $(k-1)A$'s occur in a row; *Scenario2* is that an Ω occurs after less than $(k-1)A$'s in a row.
In case of *Scenario1*, the very right τ-labeled transition is enabled, implying that the condition of the rule is satisfied. Consequently any arbitrary occurrences of A or Ω is possible. In case of *Scenario2*, after occurrence of Ω, using reset arcs

from places *Count* and P_{k-1} to Ω, the places *Count* and P_{k-1} are emptied. In this marking, there is only a token in place P_2 which enables the τ-labeled transition to the left. Firing τ brings the pattern to its initial marking with a token in place *Initial* and $(k-1)$ tokens in place P_{k-1} because the condition of the rule is not satisfied. By construction this pattern can only terminate if A occurs at least k times in a row.

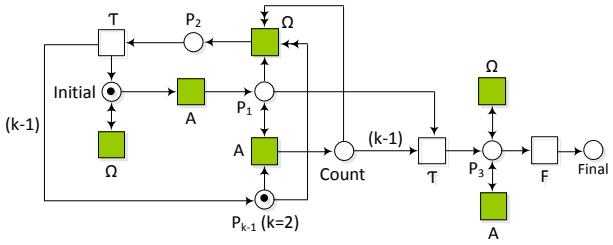

Fig. 8. 'Bounded Existence of an Event. At least k times in a row' Compliance rule

Bounded Existence of an Event. At most k times in a row .
Description: A given event A must not occur more than k times in a row (directly one after the other) within a scope. This rule is violated if A occurs more than k times in a row within the specified scope. Figure 9 shows the Petri-net pattern that formalizes this rule for the case $k = 2$.
The basic structure of this pattern is similar to the pattern described earlier in Figure 8, i.e., the reset mechanism to the initial marking is the same as in Figure 8. In the marking where there is one token in place P_1 and $(k-1)$ tokens in place P_{k-1}, both *Scenario1* and *Scenario2* (explained in the previous pattern) are possible. However based on this rule, less than k occurrences of A or exactly k occurrences of A are considered as compliant behavior. Therefore it is not necessary that all tokens in place P_{k-1} are consumed when the pattern terminates, i.e., the pattern can terminate by firing transition F_1 when there is a token in place *Initial*. Firing transition F_1 removes all possible remaining tokens in place P_{k-1} using the *reset arc* from P_{k-1} to F_1, thereby ensuring the proper completion of the pattern. Likewise the pattern can terminate when there is a token in place P_1 by firing transition F_2. Firing transition F_2 removes all possible remaining tokens in place P_{k-1} using the *reset arc* from P_{k-1} to F_2, thereby ensuring the proper completion of the pattern.

6.3 Bounded Sequence Category

This category includes compliance rules that limit the number of times a given sequence of events must or must not occur within a scope

Bounded Sequence of Events. One to one coexistence .

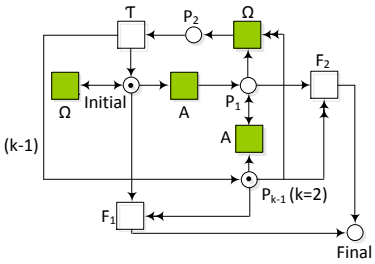

Fig. 9. 'Bounded Existence of an Event. At most k times in a row' Compliance rule

Description: For each event A there should exist one event B and for each event B there should exist one event A. If A and B do not occur in form of a pair, the rule is violated. Figure 10 shows the Petri-net pattern that formalizes this rule. Being in the initial marking, any event may occur. Occurrence of first A or B requires that the left τ-labeled transition fires. After firing τ, places P_1 and P_2 are marked. At this marking, A, B or Ω may occur in any order. Occurrence of A requires that B must follow it eventually. Symmetrically occurrence of B requires that A must follow it eventually, otherwise the right τ-labeled transition cannot fire, implying that the pattern cannot return to its initial marking. The pattern may only terminate at the initial marking by firing transition F.

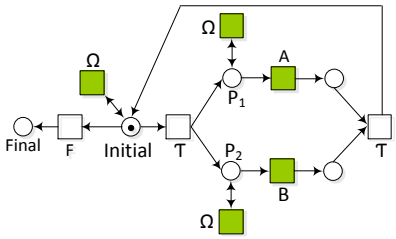

Fig. 10. 'Bounded Sequence of Events. One to one coexistence' Compliance rule

Bounded Sequence of Events. Coexistence .
Description: For any given number of events A there should exist the same number of events B. This rule is violated if the number of occurrences of A is not equal to the number of occurrences of B. Figure 11 shows the Petri-net pattern that formalizes this rule.
Being in the initial marking, any event may occur. In the upper subnet illustrated in the pattern, the place P_1 counts the occurrences of A. Each occurrence of event A increments the number of tokens in P_1, and each occurrence of event B decrements the number of tokens in P_1. This construction ensures that for each occurrence of event B, A must have occurred earlier. Therefore place P_1 becomes empty only if B occurs as many times as A has occurred.
Symmetrically in the lower subnet of the pattern, the place P_2 counts the occurrences of B. Each occurrence of B increments the number of tokens in P_2, and each occurrence of event A decrements the number of tokens in P_2. This

construction ensures that for each occurrence of event A, B must have occurred earlier. Therefore place P_2 becomes empty only if A occurs as many times as B has occurred.
The transition F models that the end of the trace has been reached. Firing transition F is only possible when two places P_1 and P_2 are empty, implying that A and B occurred the same number. The *inhibitor arcs* connecting P_1 and P_2 with transition F ensure that F can only fire if P_1 and P_2 are both empty.

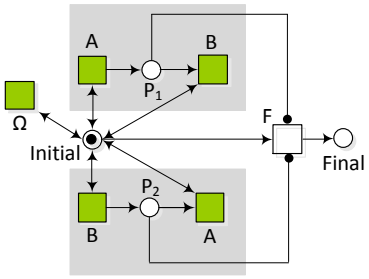

Fig. 11. 'Bounded Sequence of Events. Coexistence' Compliance rule

Bounded Sequence of Events. Exactly k times .
Description: The sequence of events $\langle A, B\rangle$ (B directly after A) must occur exactly k times within a scope. The compliance rule is violated if $\langle A, B\rangle$ does not occur k times within the specified scope. Figure 12 shows the Petri-net pattern that formalizes this rule for the case $k = 2$.
Being in the initial marking, any event may occur (please see the shadowed subnet labeled (a)). After the first occurrence of A, place P_1 is marked. A consecutive occurrence of $\langle A, B\rangle$ is modeled as a cycle. The complete cycle may occur at most k times because the preplace P_k of B is marked initially with k number of tokens. A $(k + 1)$-st occurrence of A is permitted as long as it is not followed directly by B. After A occurred there may be arbitrary occurrences of A: a subsequent B yields a direct sequence of $\langle A, B\rangle$, any other transition interrupts this sequence (Ω brings the token back to place *Initial*).
As soon as the sequence of $\langle A, B\rangle$ occurred k times, the condition of the rule is satisfied. In this situation any event may occur or the pattern can terminate by firing transition F_1.
When the place P_3 is marked with a token, any number of occurrences of B or Ω is possible but as soon as the event A occurs, the pattern should ensure that B does not occur directly after A (please see the shadowed subnet labeled (b)). Therefore if A occurs, P_4 is marked. In this marking only A or Ω may occur and B can only occur if an Ω-labeled event occurs after A.
The transitions F_1, F_2 and F_3 model that the end of the trace has been reached. The *reset arcs* connecting place *Initial* to F_1, F_2 and F_3 ensure the proper completion of the pattern.

Bounded Sequence of Events. At least k times .

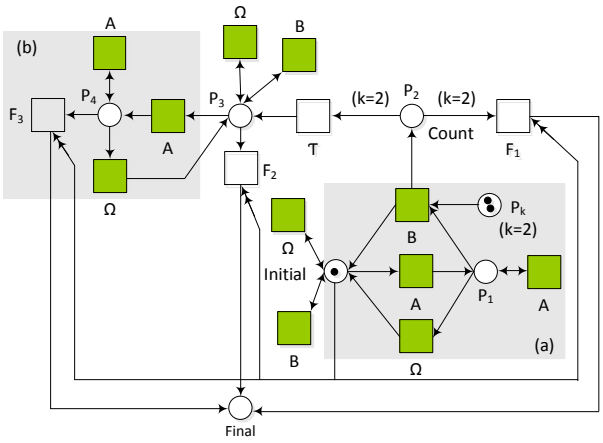

Fig. 12. 'Bounded Sequence of Events. Exactly k times' Compliance rule

Description: The sequence of events $\langle A, B\rangle$ (B exactly after A) must occur at least k times within a scope. The compliance rule is violated if sequence $\langle A, B\rangle$ occurs less than k times within the specified scope. Figure 13 shows the Petri-net pattern that formalizes this rule for the case $k = 2$.
The basic structure of this pattern is similar to the pattern described in Figure 12 (please see the shadowed subnet labeled (a) in Figure 12 and the shadowed subnet labeled (a) in Figure 13).
However this rule allows for more than k occurrences of $\langle A, B\rangle$. Therefore when the condition of rule is fulfilled, i.e., when the sequence $\langle A, B\rangle$ occurred k times, place P_2 is marked. In this situation any event may occur, even more occurrences of $\langle A, B\rangle$ is possible (please see the shadowed subnet labeled (b)). The pattern can terminate here too by firing transition F. The *reset arc* connecting place *Initial* to F ensures the proper completion of the pattern.

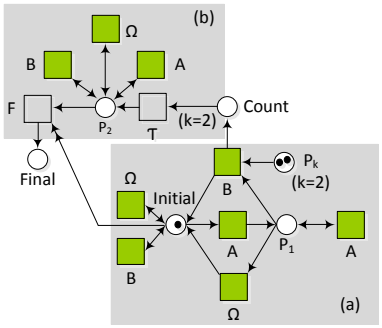

Fig. 13. 'Bounded Sequence of Events. At least k times' Compliance rule

Bounded Sequence of Events. At most k times .
Description: The sequence of events $\langle A, B\rangle$ (B exactly after A) must not occur more than k times within a scope. The compliance rule is violated if $\langle A, B\rangle$

occurs more than k times within the specified scope. Figure 14 shows the Petri-net pattern that formalizes this rule.
The basic structure of this pattern is similar to the pattern described in Figure 12 (please see the shadowed subnet labeled (a) in Figure 12).
This pattern can go to the final marking at any point when no event is to be executed, even if there are still tokens in P_k. Here, the *reset arcs* connecting P_k with the final transitions F_1 and F_2 ensure that all pending tokens are removed from the net (e.g., if $\langle A, B \rangle$ never occurred).

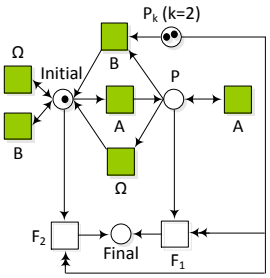

Fig. 14. 'Bounded Sequence of Events. At most k times' Compliance rule

6.4 Parallel Category

Description: A given event A must always occur in parallel with an event B. The compliance rule is violated if A and B does not occur simultaneously. This category includes one compliance rule. Figure 15 shows the Petri-net pattern that formalizes this rule.
At the beginning of this section, it was mentioned that some compliance rules require to model the start and completion of activities in the Petri-net pattern. This compliance rule requires the activity A to be represented by two events $A - start$ (A_{st}) and $A - complete$ (A_{cmp}) indicating the start and completion of A. Likewise activity B is represented by two events $B - start$ (B_{st}) and $B - complete$ (B_{cmp}).
Being in the initial marking, any event may occur. As soon as A starts, B should also start, i.e., event A cannot complete unless event B has already started. Similarly as soon as event B starts it is required that the event A also starts.
Likewise the pattern enforces by construction that the events A and B must complete together otherwise the pattern cannot return to the initial marking or terminate.
Please note that Ω-labeled event may occur independent from occurrences of A and B throughout the pattern. The transition F models that the end of the trace has been reached.

6.5 Precedence Category

This category includes compliance rules that limit the occurrence of a given event A in precedence over a given event B.

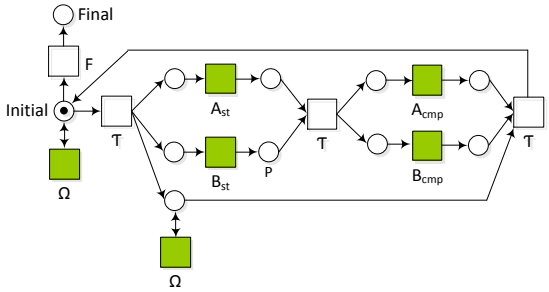

Fig. 15. 'Parallel. Parallel' Compliance rule

Precedence. Simultaneous or before .
Description: A given event A must always occur before or simultaneously with a given event B. This rule is violated if activity A occurs after activity B. The Petri-net pattern illustrated in Figure 16 formalizes this rule.
This pattern models two options (specified in the rule) for occurrences of A and B. The case where activities A and B are executed simultaneously requires that the left (τ)-labeled transition in the pattern to fire. After firing τ both A and B can start. However, B cannot complete unless A has already started. This is specified by the pre-place P of $A-complete$ (A_{cmp}). The pattern cannot return to its initial marking if both transitions A and B complete.
The case where A completes directly before B starts is also described in the main cycle of the pattern. Such that after firing the left τ, A starts and completes and directly after that B starts and completes. In this case also completion of both A and B is required such that pattern can return to its initial marking.
There is no cycle that permits the execution of activity B without a preceding or simultaneous A. If there is no B or if A just occurred, any event but B_{st} and B_{cmp} may occur. This is also the situation when the pattern may terminate by firing the transition F.

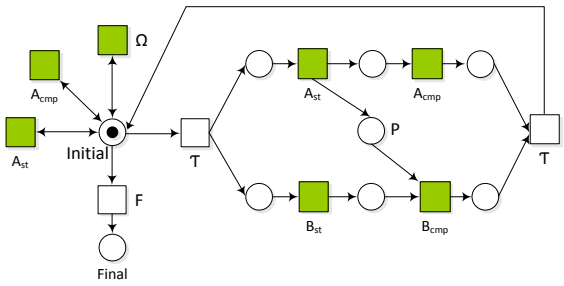

Fig. 16. 'Precedence. Simultaneous or before' Compliance rule

Precedence. Direct .
Description: Every time a given event B occurs, it should be directly preceded by an event A. If B occurs without a directly preceding A, the rule is violated. The Petri-net pattern illustrated in Figure 17 formalizes this rule.

The pattern describes a cycle of A and B, such that B can only occur if A has directly preceded it. If there is no B or A just occurred, any event may occur. This is also the situation when the pattern may terminate by firing transition F.

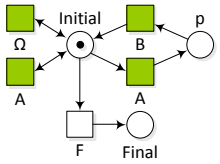

Fig. 17. 'Precedence. Direct' Compliance rule

Precedence. Direct or indirect .
Description: Every time a given event B occurs, it must be preceded (directly or indirectly) by an event A. If A does not occur before B, the rule is violated. The Petri-net pattern illustrated in Figure 18 formalizes this rule.
The pattern described in Figure 18 is similar to the pattern described in Figure 17; with the difference that the adjacent Ω-labeled transition to place P in Figure 18 allows the indirect precedence of any B with A.

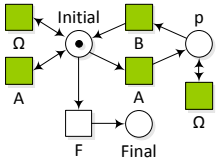

Fig. 18. 'Precedence. Direct or indirect' Compliance rule

Precedence. At least once .
Description: A given event B must be preceded by an event A at least once. If A does not precede B at least for one time, the rule is violated. The Petri-net pattern illustrated in Figure 19 formalizes this rule.
Being in the initial marking, any event but B may occur. The pattern structure describes that B can only occur if before that at least one time A has occurred. As soon as A occurs, place P is marked. In this marking any event may occur because the condition of the rule is satisfied. B may occur arbitrary numbers and in any order with respect to A and Ω. The transitions F_1 and F_2 model that the end of the trace has been reached, if no event is to be executed.

Precedence. Direct multiple events .
Description: Every time a given event B occurs, it must be preceded directly by event B or A. If directly before B one of the events B or A does not occur, the rule is violated. The Petri-net pattern illustrated in Figure 20 formalizes this rule.

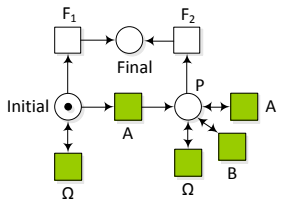

Fig. 19. 'Precedence. At least once' Compliance rule

The pattern structure describes that B can only occur if directly before it any of the events A or B has occurred. Being in the initial marking, any event but B may occur because before the first B at least once A must have occurred. After the first occurrence of A, place P is marked. In this marking, A or B can occur in any order and still the condition of the rule is satisfied. The transitions F_1 and F_2 model that the end of the trace has been reached, if no event is to be executed.

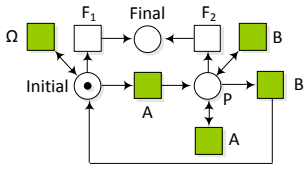

Fig. 20. 'Precedence. Direct multiple events' Compliance rule

Precedence. Direct or indirect multiple events .
Description: Every time a given event B occurs, it must be preceded by event B or event A. If before B one of the events B or A does not occur, the rule is violated. The Petri-net pattern illustrated in Figure 21 formalizes this rule.
The pattern in Figure 21 is similar to the pattern described in Figure 20; with the difference that as long as B is preceded (even indirectly) by any of the events A or B, the condition of the rule is satisfied. The adjacent Ω-labeled transition to place P allows for indirect precedence of B by A or B. The transitions F_1 and F_2 model that the end of the trace has been reached, if no event is to be executed.

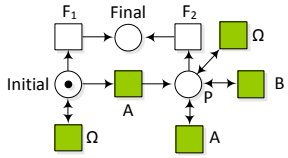

Fig. 21. 'Precedence. Direct or indirect multiple events' Compliance rule

Precedence. Direct multiple different events .

Description: Every time a given event B occurs, it should be directly preceded by C or A. If directly before B one of the events A or C does not occur, the rule is violated. The Petri-net pattern illustrated in Figure 22 formalizes this rule.
Being in the initial marking, any event but B may occur. Event B can occur only when place P is marked, implying that A or C has already occurred. The transitions F_1 and F_2 model that the end of the trace has been reached, if no event is to be executed.

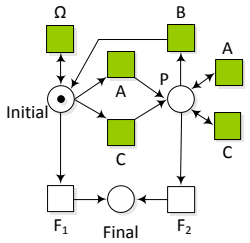

Fig. 22. 'Precedence. Direct multiple different events' Compliance rule

Precedence. Direct or indirect multiple different events .
Description: Every time a given event B occurs, it should be preceded at least once by event C or event A. If before B one of the events A or C does not occur, the rule is violated. The Petri-net pattern illustrated in Figure 23 formalizes this rule.
The pattern in Figure 23 is similar to the pattern described in Figure 22; with the difference that as long as B is preceded (even indirectly) by any of the events A or C, the condition of the rule is satisfied. The adjacent Ω-labeled transition to place P allows for indirect precedence of B by A or C. The transitions F_1 and F_2 model that the end of the trace has been reached, if no event is to be executed.

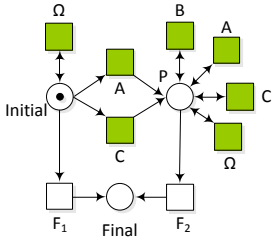

Fig. 23. 'Precedence. Direct or indirect multiple different events' Compliance rule

Precedence. Never direct .
Description: A given event B must never be preceded directly by A. If A occurs directly before B, the rule is violated. The Petri-net pattern illustrated in Figure 24 formalizes this rule.

Being in the initial marking, any event may occur. As soon as A occurs, the structure of the pattern should ensure that B cannot occur directly after A. Therefore after A occurs, place P is marked and B is not enabled anymore. B may occur only after occurrence of an Ω. The pattern can terminate at any point by firing any of the transitions F_1 and F_2, if no event is to be executed.

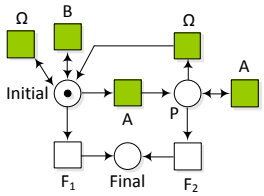

Fig. 24. 'Precedence. Never direct' Compliance rule

Precedence. Never .
Description: A given event B must never be preceded by A. If A even once occurs before B, the rule is violated. The Petri-net pattern illustrated in Figure 25 formalizes this rule.
Being in the initial marking any event may occur. As soon as A occurs, the structure of the pattern should ensure that B cannot occur anymore. Therefore after A occurs, place P is marked and B is not enabled anymore. The pattern can terminate at any point by firing any of the transitions F_1 and F_2, if no event is to be executed.

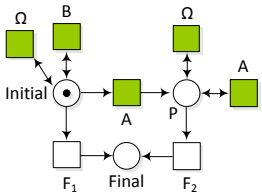

Fig. 25. 'Precedence. Never' Compliance rule

6.6 Chain Precedence Category

A sequence of events $\langle B_1, \ldots, B_m \rangle$ must or must not be preceded directly or indirectly by a sequence of events $\langle A_1, \ldots, A_n \rangle$.

Chain Precedence. Direct .
Description: Every sequence of events $\langle B_1, B_2 \ldots, B_m \rangle$ must be preceded directly by a sequence of events $\langle A_1, A_2, \ldots, A_n \rangle$. The rule is violated if directly before the sequence $\langle B_1, B_2, \ldots, B_m \rangle$, the sequence $\langle A_1, A_2, \ldots, A_n \rangle$ does not occur. The Petri-net pattern illustrated in Figure 26 formalizes this rule.
This pattern describes the allowed behaviors specified in the rule in two cycles. In the main cycle of the pattern (please see the shadowed subnet labeled (a)),

it is specified that the sequence $\langle B_1, B_2, \ldots, B_m \rangle$ can only occur if the sequence $\langle A_1, A_2, \ldots, A_n \rangle$ has occurred directly before it. From any place in the main cycle (subnet(a)) where any of the sequences $\langle A_1, A_2, \ldots, A_n \rangle$ or $\langle B_1, B_2, \ldots, B_m \rangle$ does not complete, it is possible to return to the initial marking or terminate the pattern if no event is to be executed. The return paths are indicated by the smaller cycles inside the main cycle of the pattern.
Being in the initial marking, occurrence of any sequence over the set of events $\Sigma_L \setminus \{A_1, B_1\}$ is possible and the pattern can also terminate in this situation if it reaches its end. However as soon as A_1 occurs, its occurrence is captured in the main cycle of the pattern in order to provide the possibility to detect the behavior if $\langle A_1, A_2, \ldots, A_n \rangle$ completes.
Likewise as soon as B_1 occurs the left cycle (please see the shadowed subnet labeled (b)) in the pattern is followed, in order to avoid the completion of the sequence $\langle B_1, B_2, \ldots, B_m \rangle$. In this cycle, at most the occurrence of the sequence $\langle B_1, B_2, \ldots, B_{m-1} \rangle$ is possible. From any place in this cycle where the sequence $\langle B_1, B_2, \ldots, B_{m-1} \rangle$ does not complete, it is possible to return to the initial marking or terminate the pattern if no event is to be executed.

Chain Precedence. Direct or indirect .
Description: Every sequence of events $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ must be preceded by the sequence of events $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$. The rule is violated if before the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$, the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ does not occur. The Petri-net pattern illustrated in Figure 27 formalizes this rule.
The behavior of this pattern is similar to the pattern described in Figure 26, with the difference that both direct or indirect precedence of the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ by the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ considered to be compliant (based on the compliance rule).
The pattern in Figure 27 describes the allowed behaviors specified in the rule in two cycles. In the main cycle of the pattern (please see the shadowed subnet labeled (a)), it is specified that the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ can only occur if the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ has occurred before it. From any place between P_1 to P_n in the main cycle, where the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ does not complete, it is possible to return to the initial marking or terminate the pattern if no event is to be executed.
If the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ completes, P_n in the main cycle is marked. At this marking, it is possible to execute any event, implying the possibility of indirect precedence of the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ by the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$.
However as soon as B_1 occurs, the token is removed from place P_n, in order to provide the possibility to detect the behavior in case $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ completes. From any place after transition B_1 in the main cycle where $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ is not completed, it is possible to return to the place P_n or terminate the pattern if no event is to be executed. However as soon as $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ completes, the pattern returns to its initial marking in order to satisfy the compliance rule if the next sequence of $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ occurs.

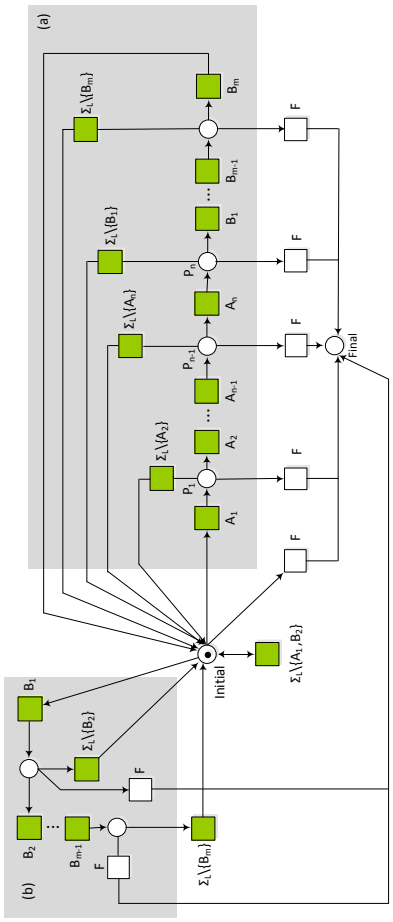

Fig. 26. 'Chain Precedence. Direct' Compliance rule

Being in the initial marking, occurrence of any sequence over the set of events $\Sigma_L \setminus \{A_1, B_1\}$ is possible and the pattern can also terminate in this situation if it reaches its end. However as soon as A_1 occurs, its occurrence is captured in the main cycle of the pattern in order to provide the possibility to detect the behavior if $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ completes. Likewise as soon as B_1 occurs the left cycle (please see the shadowed subnet labeled (b)) in the pattern is followed, in order to avoid the completion of the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$. In this cycle, at most the occurrence of the sequence $\langle B_1, \ldots, B_2, \ldots, B_{m-1} \rangle$ is possible. From any place after transition B_1 in this cycle where the sequence $\langle B_1, \ldots, B_2, \ldots, B_{m-1} \rangle$ does not complete, it is possible to return to the initial marking or terminate the pattern if no event is to be executed.

Please note that Ω-labeled event may occur any time throughout the entire pattern, even within the specified sequences of the rule: $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ and $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$.

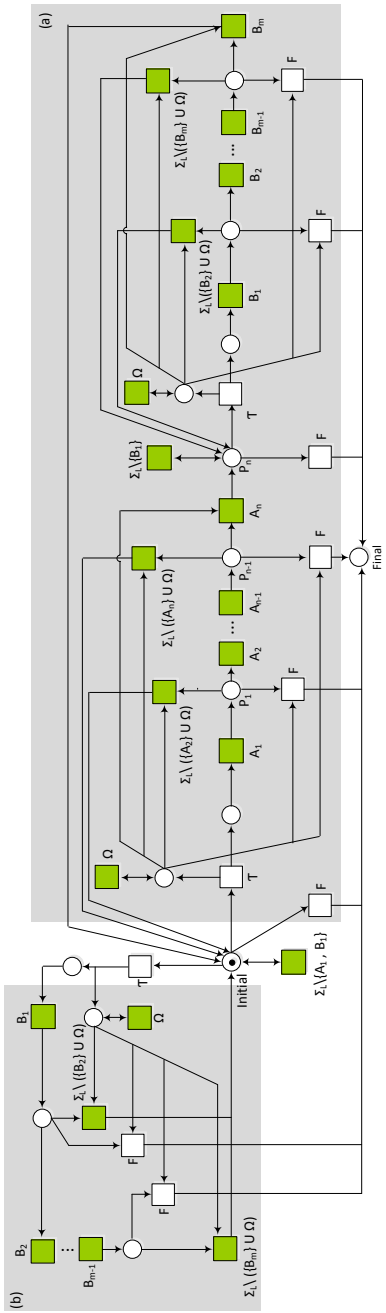

Fig. 27. 'Chain Precedence. Direct or indirect' Compliance rule

Chain Precedence. Never direct .

Description: A given sequence of events $\langle B_1, B_2, \ldots, B_m \rangle$ must not be preceded directly by a sequence of events $\langle A_1, A_2, \ldots, A_n \rangle$. The rule is violated if directly before the sequence $\langle B_1, B_2, \ldots, B_m \rangle$, the sequence $\langle A_1, A_2, \ldots, A_n \rangle$ occurs. The Petri-net pattern illustrated in Figure 28 formalizes this rule.
In the main cycle of the pattern, it is specified that the sequence $\langle B_1, B_2, \ldots, B_m \rangle$ cannot occur if the sequence $\langle A_1, A_2, \ldots, A_n \rangle$ has occurred directly before it. This is ensured by the last transition in the main cycle, being any transition but B_m; implying that the sequence $\langle B_1, B_2, \ldots, B_m \rangle$ can never complete directly after the sequence $\langle A_1, A_2, \ldots, A_n \rangle$. From any place in the main cycle, where any of the sequences $\langle A_1, A_2, \ldots, A_n \rangle$ or $\langle B_1, B_2, \ldots, B_{m-1} \rangle$ does not complete, it is possible to return to the initial marking or terminate the pattern, if no event is to be executed. The return paths are indicated with smaller cycles inside the main cycle of the pattern.
Being in the initial marking, occurrence of any sequence over the set of events $\Sigma_L \setminus \{A_1\}$ is possible. However as soon as A_1 occurs, its occurrence is captured in the main cycle of the pattern in order to provide the possibility to detect the behavior if $\langle A_1, A_2, \ldots, A_n \rangle$ completes.
The pattern can terminate at any point in time if it reaches its end and no event is to be executed.

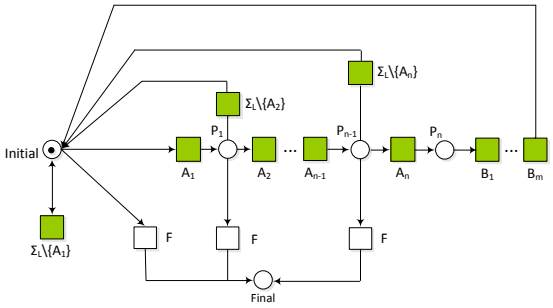

Fig. 28. 'Chain Precedence. Never direct' Compliance rule

Chain Precedence. Never .
Description: A given sequence of events $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ must never be preceded by a sequence of events $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$. The rule is violated if any time before occurrence of the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$, the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ occurs. The Petri-net pattern illustrated in Figure 29 formalizes this rule.
The behavior of this pattern is similar to the pattern described in Figure 28, with the difference that the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ can never (neither directly nor indirectly) be preceded by the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$. This is ensured in the pattern by the last transition in the main cycle, being any transition but B_m or Ω; implying that the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ can never complete after the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ occurred. In the main cycle as soon as $\langle A_1, \ldots A_2, \ldots, A_n \rangle$ completes, the place P_n is marked. At this

marking any event may occur as long as the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ does not complete. Moreover after this marking, the cycle cannot return to the initial marking to ensure that the sequence $\langle B_1, \ldots B_2, \ldots, B_m \rangle$ can never occur (even indirectly) after the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$. The pattern can terminate at any point in time if it reaches its end and no event is to be executed.
Please note that Ω-labeled event may occur any time throughout the entire pattern, even within the specified sequences of the rule: $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ and $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$.
Being in the initial marking, occurrence of any sequence over the set of events $\Sigma_L \setminus \{A_1\}$ is possible and the pattern can also terminate in this situation if it reaches its end. As soon as A_1 occurs its occurrence is captured in the main cycle of the pattern in order to provide the possibility to detect the behavior if $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ completes.

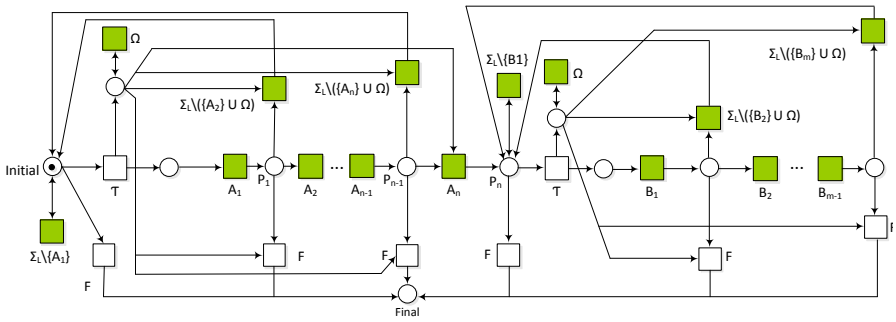

Fig. 29. 'Chain Precedence. Never' Compliance rule

6.7 Response Category

This category includes compliance rules that limit the occurrence of a given event B in response to an event A.

Response. Simultaneous or after .
Description: A given event A must be followed directly by event B or it must occur simultaneously with event B. If B does not occur directly after A or simultaneously with A, the rule is violated. The Petri-net pattern illustrated in Figure 30 formalizes this rule.
The pattern illustrated in Figure 30 is similar to the pattern described in Figure 16; with the difference in adjacent transitions to the place *Initial*. Such that similar to the pattern described in Figure 16, the pattern illustrated in Figure 30 models two options (specified in the rule) for occurrences of A and B. The case where events A and B occur simultaneously and the case where B starts directly after A is completed. Both cases are described in the main cycle of the pattern. However in the current compliance rule the execution of A restricts the behavior of the pattern. Therefore B and (Ω)-labeled transitions are adjacent to the place *Initial* implying; being in the initial marking, if there is no A, B

or any (Ω)-labeled transitions may occur. This is also the situation when the pattern may terminate by firing transition F.

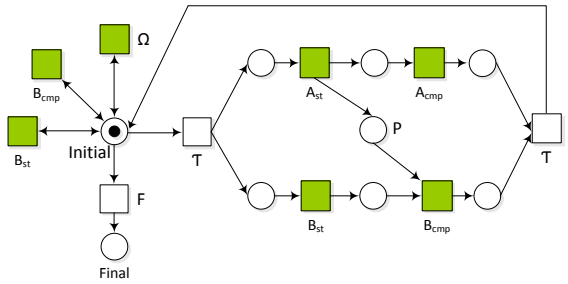

Fig. 30. 'Response. Simultaneous or after' Compliance rule

Response. Direct .
Description: Every time a given event A occurs, it must be followed directly by event B. If B does not occur directly after A, the rule is violated. The Petri-net pattern illustrated in Figure 31 formalizes this rule.
The pattern described in Figure 31 is similar to the pattern described in Figure 17; with the difference in adjacent transitions to the place *Initial*. In the current pattern occurrence of event A restricts the behavior of the pattern. Therefore B and (Ω)-labeled transitions are adjacent to the place *Initial* implying; being in the initial marking, if there is no A, B or any (Ω)-labeled transitions may occur. This is also the situation when the pattern may terminate by firing transition F.

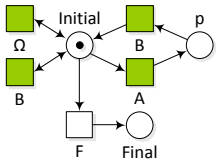

Fig. 31. 'Response. Direct' Compliance rule

Response. Direct or indirect .
Description: Every time a given event A occurs, it must be followed eventually by event B. If B does not occur after A, the rule is violated. The Petri-net pattern illustrated in Figure 32 formalizes this rule.
The pattern illustrated in Figure 32 is similar to the pattern described in Figure 31; with the difference that the adjacent Ω-labeled transition to the place P, allows that B indirectly follows any A.

Response. At least once .

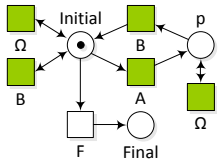

Fig. 32. 'Response. Direct or indirect' Compliance rule

Description: A given event A must always be followed eventually by B. If B does not occur at least once after A, the rule is violated. The Petri-net pattern illustrated in Figure 33 formalizes this rule.

The pattern structure describes that A can only occur if after it, at least one time B occurs. The pattern illustrated in Figure 33 is similar to the pattern described in the Figure 32; with the difference that the adjacent A-labeled transition to place P in Figure 33, allows for arbitrary numbers of occurrences of A. However, eventually B must occur to satisfy the condition of the rule.

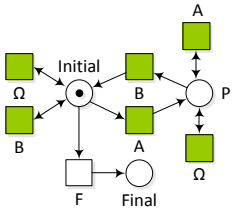

Fig. 33. 'Response. At least once' Compliance rule

Response. Direct multiple events .

Description: Every time a given event A occurs, it must be followed directly by event B or event A. If directly after A one of the events B or A does not occur, the rule is violated. The Petri-net pattern illustrated in Figure 34 formalizes this rule.

The pattern structure describes that A can only occur if directly after it any of the events A or B occurs. Being in the initial marking, any event may occur. As soon as A occurs, the place P is marked. In this marking A may occur an arbitrary number, but the only possibility to return to the initial marking is the occurrence of B. Only in this situation the pattern can terminate by firing transition F.

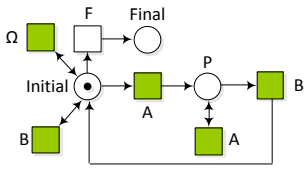

Fig. 34. 'Response. Direct multiple events' Compliance rule

Response. Indirect multiple events .
Description: Every time a given event A occurs, it must be followed eventually by event B or event A. If after A one of the events B or A does not occur, the rule is violated. The Petri-net pattern illustrated in Figure 35 formalizes this rule.
The pattern illustrated in Figure 35 is similar to the pattern described in Figure 34; with the difference that as long as A is followed (even indirectly) by any of the events A or B, the condition of the rule is satisfied. The adjacent Ω-labeled transition to the place P allows that A is followed indirectly by A or B.

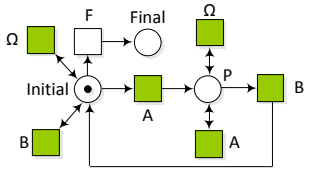

Fig. 35. 'Response. Indirect multiple events' Compliance rule

Response. Direct multiple different events .
Description: Every time a given event A occurs, it must be followed directly by event B or event C. If directly after A one of the events B or C does not occur, the rule is violated. The Petri-net pattern illustrated in Figure 36 formalizes this rule.
Being in the initial marking, any event may occur. This pattern describes two options (specified in the rule) for occurrence of A. The case where B directly follows A is formalized by the upper cycle of the net and the case where C directly follows A is formalized by the lower cycle of the net. There is no cycle that permits an occurrence of event A without a following B or C. The transition F models that the end of the trace has been reached, if no event is to be executed.

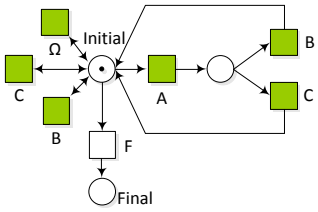

Fig. 36. 'Response. Direct multiple different events

Response. Indirect multiple different events .
Description: Every time a given event A occurs, it must be followed at least once eventually by event B or event C. If B or C does not occur at least one time after A, the rule is violated. The Petri-net pattern illustrated in Figure 37 formalizes this rule.

The pattern illustrated in Figure 37 is similar to the pattern described in Figure 36; with the difference that as long as A is followed (even indirectly) by any of the events B or C, the condition of the rule is satisfied. The adjacent Ω-labeled transition to the place P allows that A is followed indirectly by B or C. The transition F models that the end of the trace has been reached, if no event is to be executed.

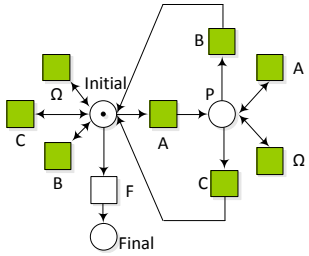

Fig. 37. 'Response. Indirect multiple different events' Compliance rule

Response. Never direct .
Description: A given event A must never be followed directly by event B. If B occurs directly after A, the rule is violated. The Petri-net pattern illustrated in Figure 38 formalizes this rule.
Being in the initial marking, any event may occur. As soon as A occurs, the structure of the pattern should ensure that B cannot occur directly after A. Therefore after A occurs, place P is marked and B is not enabled anymore. B may occur only after occurrence of an Ω. The pattern can terminate at any point by firing any of the transitions F_1 and F_2, if no event is to be executed.

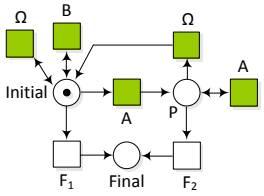

Fig. 38. 'Response. Never direct' Compliance rule

Response. Never .
Description: A given event A must never be followed by B. If any time B occurs after A, the rule is violated. The Petri-net pattern illustrated in Figure 39 formalizes this rule.
Being in the initial marking any event may occur. As soon as A occurs, the structure of the pattern should ensure that B cannot occur anymore. Therefore after A occurs, place P is marked and B is not enabled anymore. The pattern can terminate at any point by firing any of the transitions F_1 and F_2, if no event is to be executed.

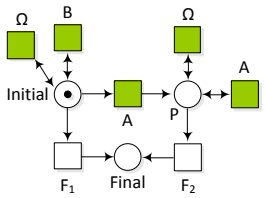

Fig. 39. 'Response. Never' Compliance rule

6.8 Chain Response Category

A sequence of events $\langle A_1, A_2 \ldots, A_n \rangle$ must or must not be followed directly or indirectly by a sequence of events $\langle B_1, B_2, \ldots, B_m \rangle$.

The compliance patterns in this category are similar to the compliance patterns described in the category *Chain Response Category* in Section 6.6, with slight differences in termination of the patterns and the adjacent transitions to the place *Initial*. In the patterns described in Section 6.6, the occurrence of the sequence of events $\langle B_1, B_2, \ldots, B_m \rangle$ puts limitation on the behavior of the patterns; while in the patterns described in the current category *Chain Response Category*, the occurrence of sequence of events $\langle A_1, A_2 \ldots, A_n \rangle$ limits the compliant behavior of the patterns.

Chain Response. Direct .
Description: Every sequence of events $\langle A_1, A_2, \ldots, A_n \rangle$ must be followed directly by a sequence of events $\langle B_1, B_2, \ldots, B_m \rangle$. The rule is violated if directly after the sequence $\langle A_1, A_2, \ldots, A_n \rangle$, the sequence $\langle B_1, B_2, \ldots, B_m \rangle$ does not occur. The Petri-net pattern illustrated in Figure 40 formalizes this rule.
In the main cycle of the pattern it is specified that the sequence $\langle A_1, A_2, \ldots, A_n \rangle$ can only occur if it is followed directly by the sequence $\langle B_1, B_2, \ldots, B_m \rangle$. From any place in the main cycle, between place *Initial* and place P_{n-1} where the sequence $\langle A_1, A_2, \ldots, A_n \rangle$ does not complete, it is possible to return to the initial marking or terminate the pattern if no event is to be executed. The return paths are indicated with smaller cycles inside the main cycle of the pattern. As soon as the sequence $\langle A_1, A_2, \ldots, A_n \rangle$ is complete, the sequence $\langle B_1, B_2, \ldots, B_m \rangle$ must occur directly after that.
Being in the initial marking, occurrence of any sequence over the set of events $\Sigma_L \setminus \{A_1\}$ is possible and the pattern can also terminate in this situation if it reaches its end. As soon as A_1 occurs its occurrence is captured in the main cycle of the pattern in order to provide the possibility to detect the behavior if $\langle A_1, A_2, \ldots, A_n \rangle$ completes.

Chain Response. Direct or indirect .
Description: Every sequence of events $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ must be followed eventually by a sequence of events $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$. The rule is violated if after the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$, the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ does not occur. The Petri-net pattern illustrated in Figure 41 formalizes this rule.

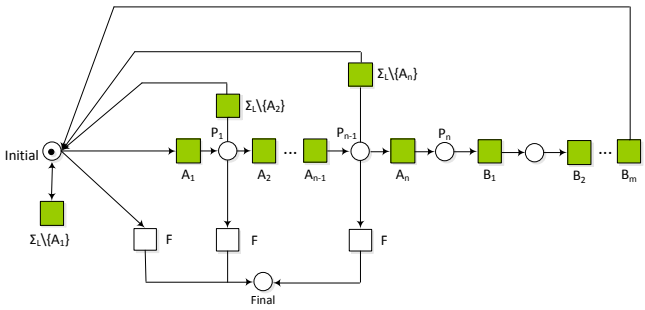

Fig. 40. 'Chain Response. Direct' Compliance rule

The behavior of this pattern is similar to the pattern described in Figure 40, with the difference that both indirect or direct occurrence of the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ after the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ considered to be compliant based on the compliance rule.

In the main cycle of the pattern, it is specified that the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ can only occur if it is followed eventually by the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$. From any place in the main cycle between place *Initial* to P_{n-1} where the $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ does not complete, it is possible to return to the initial marking or terminate the pattern if no event is to be executed. The return paths are indicated with smaller cycles inside the main cycle of the pattern. As soon as $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ completes, place P_n is marked. At this marking any event may occur; implying the possibility that the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ can be followed indirectly by the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$. The pattern cannot terminate anymore after P_n is marked unless the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ eventually completes, however it is possible to return to the marking where P_n is marked.

Please note that Ω-labeled event may occur any time throughout the entire pattern, even within the specified sequences of the rule: $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ and $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$.

Being in the initial marking, occurrence of any sequence over the set of events $\Sigma_L \setminus \{A_1\}$ is possible and the pattern can also terminate in this situation if it reaches its end. As soon as A_1 occurs its occurrence is captured in the main cycle of the pattern in order to provide the possibility to detect the behavior if $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ completes.

Chain Response. Never direct .

Description: A given sequence of events $\langle A_1, A_2, \ldots, A_n \rangle$ must never be followed directly by a sequence of events $\langle B_1, B_2, \ldots, B_m \rangle$. The rule is violated if directly after the sequence $\langle A_1, A_2, \ldots, A_n \rangle$, the sequence $\langle B_1, B_2, \ldots, B_m \rangle$ occurs. The Petri-net pattern illustrated in Figure 42 formalizes this rule.

In the main cycle of the pattern it is specified that the sequence $\langle B_1, B_2, \ldots, B_m \rangle$ cannot occur directly after the sequence $\langle A_1, A_2, \ldots, A_n \rangle$ has occurred. This is

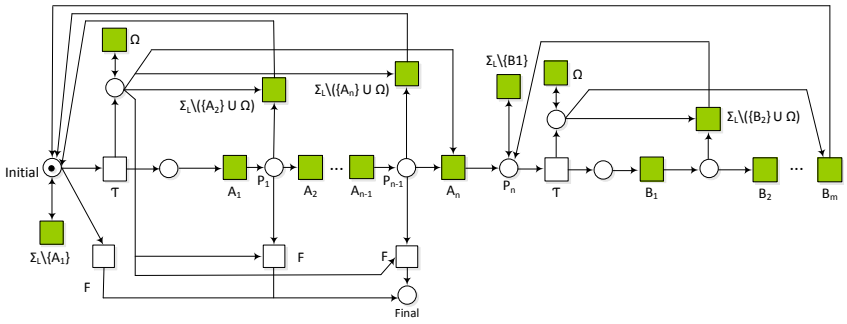

Fig. 41. 'Chain Response. Direct or indirect' Compliance rule

ensured by the last transition in the main cycle, being any transition but B_m; implying that the sequence $\langle B_1, B_2, \ldots, B_m \rangle$ can never complete directly after the sequence $\langle A_1, A_2, \ldots, A_n \rangle$. From any place in the main cycle, where any of the sequences $\langle A_1, A_2, \ldots, A_n \rangle$ or $\langle B_1, B_2, \ldots, B_{m-1} \rangle$ does not complete, it is possible to return to the initial marking. The pattern can terminate at any point in time if it reaches its end and no event is to be executed.

The remaining structure of this pattern is already explained in the pattern described in Figure 40.

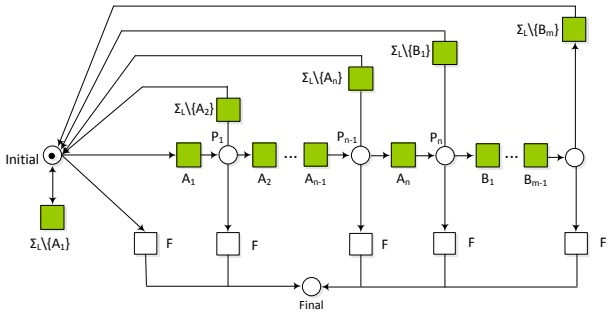

Fig. 42. 'Chain Response. Never direct' Compliance rule

Chain Response. Never .

Description: A given sequence of events $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ must never be followed by a sequence of events $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$. The rule is violated if any time after the occurrence of the sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$, the sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ occurs. The Petri-net pattern illustrated in Figure 29 formalizes the current compliance rule and the *Chain Precedence. Never* compliance rule.

The behavior of the pattern is already described in Section 6.6.

6.9 Between Category

The compliance rules in this category limit the occurrence of a given event B between a sequence of events $\langle A, \ldots, C \rangle$.

Between. After-Before .
Description: Every event B must always occur after an occurrence of event A and before an occurrence of event C. The rule is violated if B does not occur between A and C (after A and before C). The Petri-net pattern illustrated in Figure 43 formalizes this rule.
Being in the initial marking, any event but B may occur. B can only occur if A has already occurred before it. Moreover B must be followed eventually by C, otherwise there is no possibility to return to the initial marking; implying that the pattern cannot terminate as well.

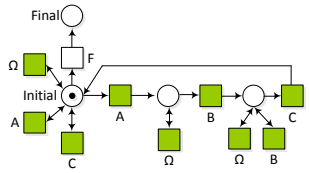

Fig. 43. 'Between. After-Before' Compliance rule

Between. Simultaneously or after-Before .
Description: Every event B must always occur directly after or simultaneously with an occurrence of event A and directly before an occurrence of event C. The rule is violated if B does not occur after or simultaneous with A and directly before C. The Petri-net pattern illustrated in Figure 44 formalizes this rule.
Being in the initial marking, any event but B may occur. The compliance rule specifies two possibilities for occurrence of B: i) simultaneously with A and directly before C, ii) directly after A and directly before C.
The occurrence of B with respect to A (simultaneous with A or directly after A) is described in the shadowed subnet in Figure 44, which is similar to the structure already described in the pattern in Figure 16.
After the completion of B, the place P is marked. At this marking C must occur, otherwise there is no possibility to return to the initial marking; implying that the pattern cannot terminate. The transition F models that the end of the trace has been reached, if no event is to be executed.

Between. After-Simultaneously or before .
Description: Every event B must always occur directly after an occurrence of event A and directly before or simultaneously with an occurrence of event C. The rule is violated if B does not occur directly after A and directly before or simultaneously with C. The Petri net pattern illustrated in Figure 45 formalizes this rule.

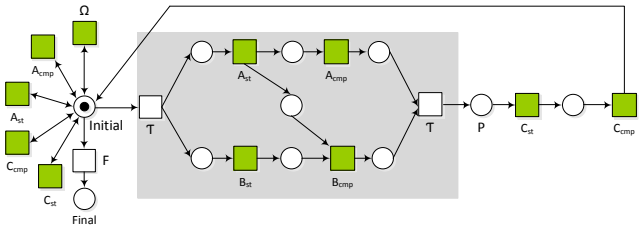

Fig. 44. 'Between. Simultaneously or after-Before' Compliance rule

Being in the initial marking, any event but B may occur. After A has occurred, the compliance rule specifies two possibilities for occurrence of B: i) directly after A and simultaneous with C, ii) directly after A and directly before C.
The occurrence of B with respect to C, (simultaneous with C or directly before C) is described in the shadowed subnet in Figure 45, which is similar to the structure already described in the pattern in Figure 16.
The pattern can return to the initial marking only if both B and C are completed. The transition F models that the end of the trace has been reached, if no event is to be executed.

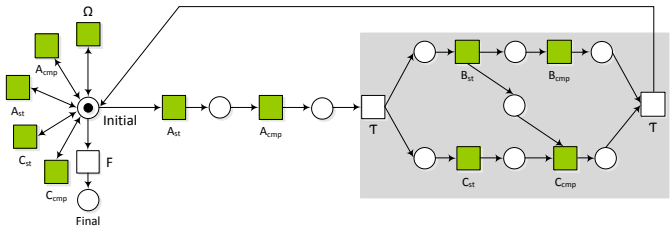

Fig. 45. 'Between. After-Simultaneously or before' Compliance rule

Between. Directly after-Directly before .
Description: Every event B must always occur directly after event A and directly before event C. The rule is violated if B does not occur between the sequence of events $\langle A, C\rangle$. The Petri-net pattern illustrated in Figure 46 formalizes this rule.
This rule specifies the occurrence of the exact sequence of events $\langle A, B, C\rangle$. Being in the initial marking, occurrence of any event but B is possible. The pattern may also terminate at this marking if no event is to be executed.

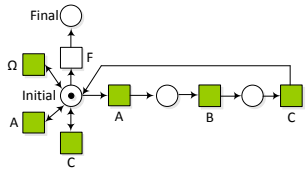

Fig. 46. 'Between. Directly after-Directly before' Compliance rule

Between. Simultaneously-Simultaneously .
Description: Every event B must always occur simultaneously with events A and C. If the event B does not occur at the same time by the events B and C, this compliance rule is violated. The Petri-net pattern illustrated in Figure 47 formalizes this rule.
Being in the initial marking, any event but B may occur. This rule specifies that as soon as B starts, the events A and C must start as well. The event B can only proceed for completion if the events A and C have already started. Consecutively the pattern can return to initial marking only if all the events A, B and C are completed. This is the situation where the pattern may terminate as well if no event is to be executed.

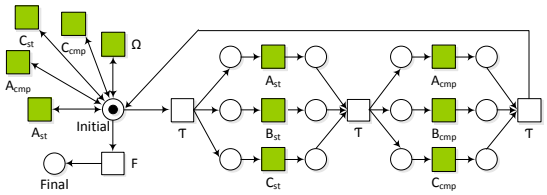

Fig. 47. 'Between. Simultaneously-Simultaneously' Compliance rule

Between. Simultaneously or after-Simultaneously or before .
Description: Every event B must always occur directly after or simultaneously with event A and directly before or simultaneously with event C. This rule is violated if B occurs before A or after C or not in the exact sequence of $\langle A, B, C\rangle$. The Petri-net pattern illustrated in Figure 48 formalizes this rule.
The pattern described in the Figure 48 is the combination of two simpler patterns described already in the Figure 16 and the Figure 30; with B being the common element in them.
The compliance rule specifies two possibilities for occurrence of B with respect to $A : i)$ directly after A, $ii)$ simultaneous with A.
The occurrence of B with respect to A (simultaneous with A or directly after A) is described in the shadowed subnet labeled (a) in Figure 48, which is similar to the structure described in the pattern in Figure 16.
Symmetrically the compliance rule specifies two possibilities for occurrence of B with respect to $C : i)$ directly before C, $ii)$ simultaneous with C.
The occurrence of B with respect to C (simultaneous with C or directly before C) is described in the shadowed subnet labeled (b) in Figure 48, which is similar to the structure described in the pattern in Figure 30.
Being in the initial marking any event may occur. This is also the situation when the pattern may terminate by firing the transition F.

Between. At least one other activity .
Description: This compliance rule specifies that between every sequence of events $\langle A, B\rangle$ or $\langle B, A\rangle$ (A before B or B before A) there should be at least one other

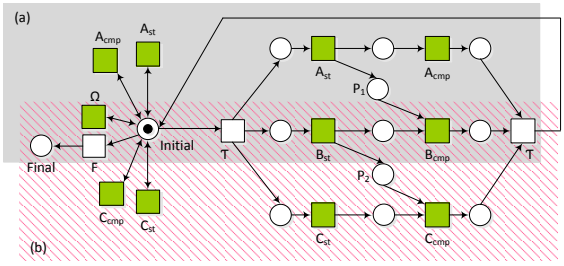

Fig. 48. 'Between. Simultaneously or after-Simultaneously or before' Compliance rule

event. This rule is violated if B occurs directly after A or if A occurs directly after B. The Petri-net pattern illustrated in Figure 49 formalizes this rule.

Being in the initial marking, any event may occur. The right cycle in the pattern ensures that as soon as event A occurs, it cannot be followed directly by B, i.e., B can only occur if after A at least one other event (Ω) is executed.

Symmetrically, the left cycle in the pattern ensures that as soon as B occurs, it cannot be followed directly by A and A can only occur if after B at least one other event (Ω) is executed.

The pattern can terminate at any point of time if no event is to be executed.

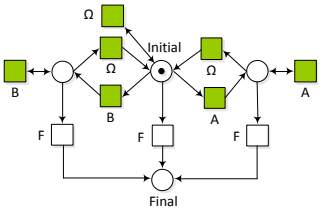

Fig. 49. 'Between. At least one other activity' Compliance rule

6.10 Exclusive Category

Description: Presence of a given event A mandates the absence of an event B. This rule is violated if both events A and B be present. This compliance category consists of one compliance rule. The Petri-net pattern illustrated in Figure 50 formalizes this rule.

Being in the initial marking, any event may occur. As soon as the first A occurs, the place P_1 is marked. At this marking B is not enabled anymore, thereby ensuring that A and B cannot be present together.

Symmetrically when the first B occurs, the place P_2 is marked. At this marking A is not enabled anymore, thereby ensuring that A and B cannot be present together.

The pattern may terminate at any point in time if no event is to be executed.

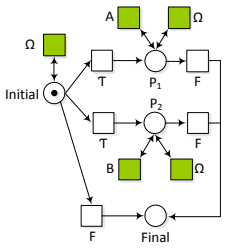

Fig. 50. 'Exclusive' Compliance rule

6.11 Mutual Exclusive Category

Description: Either a given event A or event B must exist but not none of them or both. This rule is violated if both events A and B occur together or be absent together. This compliance category consists of one compliance rule. The Petri-net pattern illustrated in Figure 51 formalizes this rule.
The behavior of this pattern is similar to the pattern described in Figure 50; with the difference that in the pattern illustrated in Figure 51 absence of both events A and B is a violation. Therefore the pattern enforces that one of the events A or B must occur. The pattern cannot terminate at initial marking too, i.e., the pattern may terminate only after the occurrence of one of the events A or B.

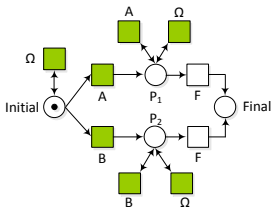

Fig. 51. 'Mutual Exclusive' Compliance rule

6.12 Prerequisite Category

Description: Absence of a given event A mandates that event B is also absent. This rule is violated if event B occurs without any occurrence of event A. This compliance category consists of one compliance rule. The Petri-net pattern illustrated in Figure 52 formalizes this rule.
Being in the initial marking, any event may occur. If A occurs, it may be followed by B or not. In both cases, the behavior is compliant i.e., presence of A does not oblige anything. The pattern may terminate in this situation if no event is to be executed. However as soon as B occurs the structure of the pattern must ensure that A also occurs at least once. Therefore occurrence of B requires that the left τ-labeled transition has already fired before it, implying that the places P_1 and P_2 are marked. At this marking both A and B are enabled and the pattern cannot return to its initial marking unless both A and B occur. The

next occurrences of B (even if they are not followed by A) are still compliant as A has already occurred once and the rule is satisfied.

The pattern may terminate at initial marking as well if it reaches its end; absence of both events is allowed based on the rule.

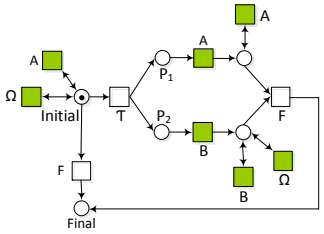

Fig. 52. 'Prerequisite' Compliance rule

6.13 Inclusive Category

Description: Presence of a given event A mandates that event B is also present. This rule is violated if event A occurs without any occurrence of event B. This compliance category consists of one compliance rule. The Petri-net pattern illustrated in Figure 53 formalizes this rule.

The pattern described in Figure 53 is similar to the pattern described in Figure 52; with the difference in adjacent transitions to the place *Initial*.

Being in the initial marking, any event may occur. If B occurs, it may be followed by B or not. In both cases, the behavior is compliant i.e., presence of B does not oblige anything.

As soon as the first A occurs, the structure of the pattern must ensure that B also occurs at least once. Next occurrences of A (even without a following B) will be still compliant as B has already occurred once, so the condition of the rule is satisfied. This is the situation that the pattern can terminate if no event is to be executed.

Please note that the pattern may terminate at initial marking as well, because based on the compliance rule it is possible that B occurs without occurrence of A or none of A or B occurs.

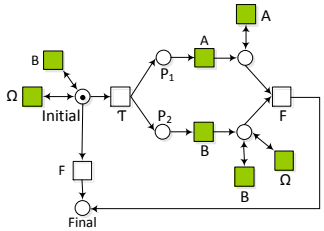

Fig. 53. 'Inclusive' Compliance rule

6.14 Substitute Category

Description: A given event B substitutes the absence of event A. This rule is basically the logical OR between occurrences of two events A and B. This rule is violated if non of the events A or B occur (i.e., both be absent). This compliance category consists of one compliance rule. The Petri-net pattern illustrated in Figure 54 formalizes this rule.
Being in the initial marking, any event may occur. The pattern cannot terminate at this marking because at least one of the events A or B is required to occur. The occurrence of A does not oblige the occurrence or non-occurrence of B, however its absence obliges the occurrence of B. The pattern can only terminate by firing the transition F; implying that at least one of the events A or B has occurred.

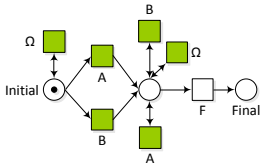

Fig. 54. 'Substitute' Compliance rule

6.15 Co-requisite Category

Description: Either given events A and B should exist together or be absent together. This rule is violated if only one of the events A or B occurs. This compliance category consists of one compliance rule. The Petri-net pattern illustrated in Figure 55 formalizes this rule.
The pattern described in Figure 55 is similar to the pattern described in Figure 52; with the difference in adjacent transitions to the place *Initial*.
Being in the initial marking, any event may occur. The pattern may terminate at this marking as well, because based on the rule absence of both events A and B is compliant. As soon as event A occurs, the structure of the pattern must ensure that B also occurs. The next occurrences of A (even if they are not followed by B) are still compliant as B has occurred once and the rule is satisfied. Symmetrically if event B occurs, the structure of the pattern must ensure that A also occurs. The next occurrences of B (even if they are not followed by A) are still compliant as A has occurred once and the rule is satisfied. The pattern may terminate when both A and B are present.

7 Compliance to Data and Organizational Aspects

So far we presented a comprehensive collection of control flow compliance rules and their formalization as Petri-net patterns. These rules cover the *control flow* dimension of the compliance rule framework introduced in Fig. 1. In this section, we show how the pattern-based approach can also be applied to compliance rules

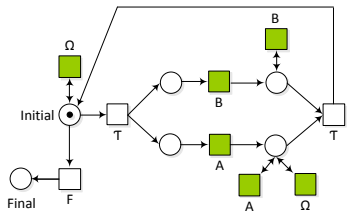

Fig. 55. 'Corequisite' Compliance rule

of Sect. 7 to check compliance with respect to *data* and to *organizational aspects*, which constitute two other dimensions of the framework. As before, we consider single-case observation-based untimed compliance rules.

7.1 Data Flow Compliance Rules

A typical example of a data flow compliance rule is to **Restrict data values permitted for a task**. For example, "A discount of 10% is granted if the customer is a gold customer; 5% are granted if the customer is a silver customer." A rule of this kind prescribes that task *grant* refers to 2 attributes e.g., *customer status* and *percentage*. When task *grant discount* occurs, these attribute values need to be logged in the corresponding event such as $(grant, John, gold, 10\%)$ (see Sect. 3); otherwise compliance cannot be checked in hindsight.

When checking compliance to this rule, it is not just sufficient to check whether *grant* occurred, but we need to check whether *grant* occurred with the right attribute values. To this end slight changes in actual Petri-net pattern and labeling ℓ that relates Petri-net transitions to events are required. Figure 56(top) shows the Petri-net pattern for this rule. It contains two transitions *grant* that are further distinguished by the attribute value combinations that are permitted by this task.

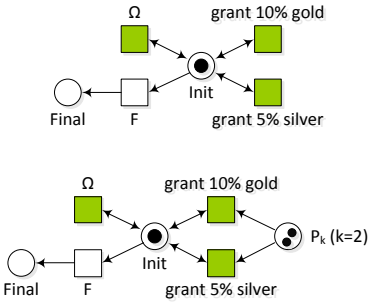

Fig. 56. Petri-net pattern to 'Restrict data values permitted for a task'.

Recall from Sect. 3 that each pattern also has a labeling function $\ell(.)$ that maps transitions to sets of events. In contrast to Sect. 6, a transition is not mapped to an event name, but to a *combination of name and attribute values*. For instance, the mapping $\ell(grant10\%gold) = \{(grant, x, y, z) \mid y = gold, z = 10\%\}$

maps transition *grant*10%*gold* only to *grant* events which have *gold* and 10% as their attribute values, correspondingly for *grant*5%*silver*.

Other occurrences of *grant* (with other attribute value combinations) are *disallowed* by mapping Ω only to events other than *grant*, e.g., $\ell(\Omega) = \{(a, x, y, \ldots) \mid a \neq grant\}$. This mapping ℓ and the pattern of Fig. 56(bottom) together formalize the compliance rule. For example, trace $\langle(add\ item, x, 10EUR)\ (add\ item, y, 32EUR)\ (grant, Joe, gold, 10\%)\rangle$ complies to this rule whereas aligning trace $\langle(add\ item, x, 10EUR)\ (add\ item, y, 32EUR)\ (grant, Jim, silver, 10\%)\rangle$ to the this rule yields a move on log $((grant, Jim, silver, 10\%), \gg)$ indicating that Jim was granted a wrong discount.

Note that data flow compliance is essentially formalized by further distinguishing transitions in the Petri-net patterns, and by defining the right mapping from transitions to events. This permits to *combine control flow rule and data flow compliance rules* also formally. For instance, the pattern of Fig. 56(bottom) formalizes that "A discount (of 10% for gold customers and 5% for silver customers) is given at most twice per case."

7.2 Compliance to Organizational Aspects

Separation of Duty (4-eyes principle). The perhaps best known compliance rule states that "Of two sequential tasks A and B, if A was performed by user R, then B must not be performed by R." Here, each task has a particular attribute *performed by* (or role) which takes as values user names or roles. Technically, the role attribute is a special data attribute: a log event $(check, Sue)$ describes that *Sue* performed activity *check*. Thus, a trace $\sigma_1 = \langle(receive, Tom)(check, Sue)(notify, Sue)(pay, Tom)\rangle$ complies to the 4-eye principle for tasks $\sigma_2 = check$ and *pay* whereas $\langle(receive, Tom)(check, Sue)(notify, Sue)(pay, Sue)\rangle$ violates the principle.

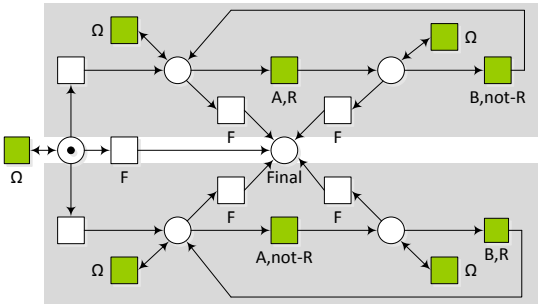

Fig. 57. Petri-net pattern for "Separation of Duty."

Figure 57 shows the Petri-net pattern that formalizes this compliance rule. It distinguishes two cases (as indicated by the upper and lower grey-shaded rectangle. Each case describes one compliant role assignment to tasks A and B, either A is performed by R, then B not by R, or vice versa. In this compliance rule, once R performed A, it always has to perform A and may never perform B (within the same trace). Hence the choice for either case is permanent in the

pattern as well. The pattern may terminate at any point in time, and all other tasks (except for A and B with the chosen role assignments) may occur at any point in time.

As for data flow compliance, the labeling $\ell(.)$ is crucial to relate patterns of the transitions to events: $\ell(A,R) = \{(x,y) \mid x = A, y = R\}, \ell(A,not\text{-}R) = \{(x,y) \mid x = A, y \neq R\}, \ell(B,R) = \{(x,y) \mid x = B, y = R\}, \ell(B,not\text{-}R) = \{(x,y) \mid x = B, y \neq R\}, \ell(\Omega) = \{(x,y) \mid x \notin \{A,B\}, y \neq R\}$. Each user gives rise to a different labeling that has to be checked separately from other labelings. When checking compliance of trace σ_2 given above w.r.t. tasks *check*, *pay*, and *Joe*, the alignment-based approach of Sect. 3 returns a move on log $((pay, Sue), \gg)$ indicating that the *pay* task should not have been performed by *Sue* (as it is not allowed by the pattern).

Altogether, compliance to data flow and to organizational aspects is orthogonal to control flow compliance and builds on mapping Petri-net transitions to events based on a combination of event name and attributes. This also allows to formalize and check rules that depend on mixture of control flow, data flow and organizational aspects.

8 Implementation in ProM

The presented technique is implemented in the *Compliance* package of the Process Mining Toolkit ProM 6, available from `http://www.processmining.org/`. The packet provides Petri-net patterns for the control flow compliance rules discussed in this report. The "*Check Compliance*" plugin takes a log as input. Then the user can pick from a list of available compliance rules, those rules against which the log shall be checked. For each rule to check, the user then configures its parameters, mostly by mapping events to task names of the rule. Then the conformance checker of Sect. 3 is called to align the log to the rule's Petri-net pattern. The resulting alignment is shown to the user. Each aligned trace is shown in a separate row and deviations are highlighted: a move on log indicates an event occurred which did not comply to the rule, a move on model indicates which event skipped in log such that log does not comply to the rule.

9 Conclusion

Today's organizations need to comply to an increasing set of laws and regulations. Compliance requirements are often described in natural language which makes checking compliance a difficult task. In this report we provided a first comprehensive collection of control flow compliance rules which allow to formally capture a large set of compliance requirements. Moreover we presented a robust technique for backwards compliance checking which enables us to provide diagnostic information in case of violations. The technique is also applicable to check compliance of artifact-centric processes [16]. The approach is supported by ProM plugins and we tested our techniques using real-life logs and compliance requirements.

The unusual choice of formalizing compliance rules as Petri-nets rather than logics posed no difficulties. Yet, we can foresee benefits from a mixed formalization of declarative rules by logics and operational rules by Petri-nets. Note that in no situation, the end user is confronted with the formalization of the rule, but picks rules by their informal description.

We showed that our approach can also handle organizational rules and data flow rules to constrain individual tasks. Handling constraints across several tasks requires to generalize the technique, in particular the underling conformance checker [8]. Also, the mapping between event attributes and transitions is cumbersome and currently specified at a technical level using concrete values; a more user-friendly approach to specify organizational and data flow rules is required.

Thus, future work aims at exploring the compliance rule framework (Fig. 1) further and extending the compliance rule set (Table 1) for other dimensions, i.e., with collections of compliance rules restricting data flow, process resource, and process time. Moreover the compliance patterns described in Section 6 follow certain systematics and a more general mechanism to produce new patterns according to these systematics is subject to future work.

Acknowledgements. We thank C. Stahl, B.F. van Dongen, and A. Adriansyah for their substantial support in this work. The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement n° 257593 (ACSI).

References

1. Aalst, W.M.P.v.d.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. Aalst, W.M.P.v.d., Adriansyah, A., Dongen, B.F.v.: Replaying history on process models for conformance checking and performance analysis. WIREs Data Mining Knowl Discov 2, 182–192 (2012)
3. Aalst, W.M.P.v.d., Beer, H.T.d., Dongen, B.F.v.: Process mining and verification of properties: An approach based on temporal logic. In: OTM Conferences 2005. LNCS, vol. 3760, pp. 130–147. Springer (2005)
4. Aalst, W.M.P.v.d., Hee, K.M.v., Werf, J.M.v.d., Kumar, A., Verdonk, M.: Conceptual Model for Online Auditing. Decision Support Systems 50(3), 636–647 (2011)
5. Abdullah, N.S., Sadiq, S.W., Indulska, M.: Information systems research: Aligning to industry challenges in management of regulatory compliance. In: PACIS 2010. p. 36. AISeL (2010)
6. Adriansyah, A., Dongen, B.F.v., Aalst, W.M.P.v.d.: Towards Robust Conformance Checking. In: BPI 2010. LNBIP, vol. 66, pp. 122–133. Springer (2011)
7. Adriansyah, A., Sidorova, N., Dongen, B.F.v.: Cost-based Fitness in Conformance Checking. In: ACSD 2011. pp. 57–66. IEEE (2011)
8. Adriansyah, A., Dongen, B.F.v., Aalst, W.M.P.v.d.: Conformance checking using cost-based fitness analysis. In: EDOC 2011. pp. 55–64. IEEE (2011)
9. Awad, A., Decker, G., Weske, M.: Efficient compliance checking using bpmn-q and temporal logic. In: BPM 2008. LNCS, vol. 5240, pp. 326–341. Springer (2008)
10. Awad, A., Weske, M.: Visualization of compliance violation in business process models. In: Business Process Management Workshops. pp. 182–193 (2009)

11. Calders, T., Guenther, C., Pechenizkiy, M., Rozinat, A.: Using Minimum Description Length for Process Mining. In: SAC 2009. pp. 1451–1455. ACM Press (2009)
12. Christopher Giblin, S.M., Pfitzmann, B.: Research report: From regulatory policies to event monitoring rules: Towards model-driven compliance automation. Tech. rep., IBM Research GmbH, Zurich Research Laboratory, Switzerland (2006)
13. Cook, J., Wolf, A.: Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. ACM Transactions on Software Engineering and Methodology 8(2), 147–176 (1999)
14. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: ICSE 1999. pp. 411–420 (1999)
15. Elgammal, A., Türetken, O., Heuvel, W.J.v.d., Papazoglou, M.P.: Root-cause analysis of design-time compliance violations on the basis of property patterns. In: ICSOC 2010. LNCS, vol. 6470, pp. 17–31 (2010)
16. Fahland, D., de Leoni, M., van Dongen, B.F., van der Aalst, W.M.P.: Conformance Checking of Interacting Processes with Overlapping Instances. In: BPM. LNCS, vol. 6896, pp. 345–361 (2011)
17. Fötsch, D., Pulvermüller, E., Rossak, W.: Modeling and verifying workflow-based regulations. In: ReMo2V 2006. CEUR Workshop Proceedings, vol. 241 (2007)
18. Ghose, A., Koliadis, G.: Auditing Business Process Compliance. In: ICSOC. Lecture Notes in Computer Science, vol. 4749, pp. 169–180 (2007)
19. Giblin, C., Liu, A.Y., Müller, S., Pfitzmann, B., Zhou, X.: Regulations expressed as logical models (realm). In: JURIX 2005. Frontiers in Artificial Intelligence and Applications, vol. 134, pp. 37–48. IOS Press (2005)
20. Goedertier, S., Martens, D., Vanthienen, J., Baesens, B.: Robust Process Discovery with Artificial Negative Events. Journal of Machine Learning Research 10, 1305–1340 (2009)
21. Gruhn, V., Laue, R.: Patterns for timed property specifications. Electr. Notes Theor. Comput. Sci. 153(2), 117–133 (2006)
22. Kharbili, M.E., Medeiros, A.K.A.d., Stein, S., Aalst, W.M.P.v.d.: Business process compliance checking: Current state and future challenges. In: MobIS 2008. LNI, vol. 141, pp. 107–113. GI (2008)
23. Kharbili, M.: Business process regulatory compliance management solution frameworks: A comparative evaluation. In: APCCM 2012. CRPIT, vol. 130, pp. 23–32. ACS (2012)
24. Lu, R., Sadiq, S.W., Governatori, G.: Compliance aware business process design. In: BBM Workshops 2007. LNCS, vol. 4928, pp. 120–131. Springer (2008)
25. Montali, M., Pesic, M., Aalst, W.M.P.v.d., Chesani, F., Mello, P., Storari, S.: Declarative Specification and Verification of Service Choreographies. ACM Transactions on the Web 4(1), 1–62 (2010)
26. Munoz-Gama, J., Carmona, J.: A Fresh Look at Precision in Process Conformance. In: BPM 2010. LNCS, vol. 6336, pp. 211–226. Springer (2010)
27. Munoz-Gama, J., Carmona, J.: Enhancing Precision in Process Conformance: Stability, Confidence and Severity. In: CIDM 2011. IEEE (2011)
28. Pitzmann, B., Powers, C., Waidner, M.: Ibm's unified governance framework (ugf). Tech. rep., IBM Research Division, Zurich (2007)
29. Ramezani, E., Fahland, D., Werf, J.M.E.M.v.d., Mattheis, P.: Separating compliance management and business process management. In: BPM Workshops 2011. LNBIP, vol. 100, pp. 459–464. Springer (2012)
30. Rozinat, A., Aalst, W.M.P.v.d.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. 33(1), 64–95 (2008)

31. Sadiq, S.W., Governatori, G., Namiri, K.: Modeling control objectives for business process compliance. In: BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer (2007)
32. Schleicher, D., Grohe, S., Leymann, F., Schneider, P., Schumm, D., Wolf, T.: An approach to combine data-related and control-flow-related compliance rules. In: SOCA 2011. pp. 1–8. IEEE (2011)
33. Schleicher, D., Anstett, T., Leymann, F., Schumm, D.: Compliant business process design using refinement layers. In: OTM Conferences 2010. LNCS, vol. 6426, pp. 114–131. Springer (2010)
34. Schleicher, D., Fehling, C., Grohe, S., Leymann, F., Nowak, A., Schneider, P., Schumm, D.: Compliance domains: A means to model data-restrictions in cloud environments. In: EDOC. pp. 257–266 (2011)
35. Schumm, D., Leymann, F., Ma, Z., Scheibler, T., Strauch, S.: Integrating compliance into business processes: Process fragments as reusable compliance controls. In: MKWI 2010. Universitätsverlag Göttingen (2010)
36. Schumm, D., Türetken, O., Kokash, N., Elgammal, A., Leymann, F., Heuvel, W.J.v.d.: Business process compliance through reusable units of compliant processes. In: ICWE Workshops 2010. LNCS, vol. 6385, pp. 325–337. Springer (2010)
37. Weerdt, J.D., Backer, M.D., Vanthienen, J., Baesens, B.: A Robust F-measure for Evaluating Discovered Process Models. In: CIDM 2011. pp. 148–155. IEEE (2011)
38. Wolter, C., Meinel, C.: An approach to capture authorisation requirements in business processes. Requir. Eng. 15(4), 359–373 (2010)