Diagnostic Information in Temporal Compliance Checking

Elham Ramezani, Dirk Fahland, Boudewijn van Dongen, and Wil van der Aalst

Eindhoven University of Technology, The Netherlands
{e.ramezani,d.fahland,b.f.v.dongen,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 an approach for checking compliance of observed process executions recorded in an event log to control-flow and temporal compliance requirements. We show a collection of 54 control flow and 15 temporal compliance rules, distributed respectively over 10 and 7 categories. In addition we present how temporal compliance requirements discussed in literature can be unified and formalized using a generic temporal compliance rule.
To check compliance with respect to a compliance rule, the event log describing the observed behavior is aligned with the corresponding rule. The alignment then shows which events occurred out of specified order and which events deviated by which amount of time from the prescribed behavior. The approach is *flexible* (easy to express new rules), and allowing for multi-perspective diagnostic information in case of compliance violations. The technique and corresponding tool support have been experimentally validated using a case study.

Keywords: compliance checking, process mining, conformance checking, data-aware 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. Five types of compliance-related activities can be identified [25, 42, 21, 39]:

- *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 compliance 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).

Compliance requirements primarily restrict sequencing of activities in the process, that is, the *control flow*; various techniques for checking control-flow compliance based on event data have been proposed including LTL-based checking.

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 [34] 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, 9, 7, 8, 12, 14, 22, 36, 37, 43, 49]. Approaches such as in [43] 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.

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 [11, 20] 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.

State-of-the-art techniques in conformance checking retrieve this information by computing *optimal alignments* [2, 9] between traces in the event log and "best fitting" paths in the model. It provides detailed diagnostic information for all deviations from

compliant behavior [41]. However, compliance requirements may also cover other aspects of a process; in particular they may restrict *process time*. Compliance violations regarding time cannot be detected using existing control-flow checking techniques.

This report addresses the problem of backward compliance checking for temporal compliance requirements (i.e., compliance requirements restricting process time). We propose a technique for temporal compliance checking that seamlessly integrates with control-flow compliance checking. Most importantly, the technique provides detailed diagnostic information in case of non-compliant behavior: it shows for each case *which* events violated temporal requirements and *when* the event should have occurred to be compliant. Our temporal compliance checking techniques leverages a recent *data-aware* conformance checking technique [29] that allows to check conformance of a log with respect to a *data-aware Petri net*. We show that every temporal compliance requirement discussed in literature (and many more) can be formalized in a simple data-aware Petri net, by making time a data attribute of the specification. The conformance checker [29] then compares the observed temporal behavior in the event data to the compliant temporal behavior specified in the Petri net. In case of deviations, the conformance checker highlights which events occurred out of order, and by how much time an event deviated from the compliant behavior. Moreover, we show how this temporal compliance checking can be combined with control-flow compliance checking to check complex compliance requirements involving control-flow and temporal aspects. The technique has been implemented as a ProM plug-in and has been validated in various real-life event data.

Moreover to address the problem of elicitation of compliance rules we provide a comprehensive collection of control flow and temporal related compliance rules. We identify 54 control-flow compliance rules distributed over 10 categories and 15 temporal compliance rules distributed over 7 categories. These compliance rules are formalized in terms of Petri-net patterns. The approach is *extendible*, i.e., to add a new type of rule, one just needs to find the relevant compliance rule and its formalization from our repository and prune it for his/her specific compliance purpose.

The remainder of this paper is organized as follows. We discuss related work in Sect. 2. We recall conformance checking techniques for control-flow and data-flow in Sect. 3. In Sec. 4 and 5 we introduce our compliance rule framework and give an overview on different categories of control-flow and temporal-compliance rules.

Sect. 6 covers control-flow compliance checking. The complete collection of control-flow compliance rules and their formalization are presented in Sect. 7.

Sect. 8 and Sect. 9 introduce the temporal compliance problem and our proposed solution. The complete collection of temporal compliance rules and their formalization are presented in Sect. 10. In Sect. 11 the implementation of the approach in ProM is showcased. Sect. 12 concludes the report.

2 Related Work

The importance of compliance management has been pointed out by various authors [5]. In [42] a life cycle is introduced to structure the process of compliance management.

A comparative analysis over different compliance management solution frameworks is provided in [25].

Compliance management has gained wide interest from the Business Process Management (BPM) community. Compliance checking approaches can be mapped onto two main categories [24]:

- *Forward* compliance checking aims at ensuring compliant process executions. Processes can be constructed to be compliant [44] or verified whether they are compliant [32]. Alternatively, compliance requirements can be transformed into monitoring rules [13] or model annotations which then are used to enforce compliant process executions [18, 51].
- *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, 9, 7, 8, 12, 14, 22, 36, 37, 43, 49]. Also approaches based on temporal logic [3, 34] have been proposed to check compliance.

Existing work in temporal compliance checking primarily focuses on verification at design time or at run time.

It is possible to derive temporal properties of acyclic process models by annotating tasks with intervals of execution and waiting times; execution times and waiting times of the entire process can then be derived by interval computations and compared against predefined constraints of total execution times [16]. In addition, the time-critical paths of a process model can be computed [40]. In a similar fashion, the approach in [31] formulates temporal constraints in terms of deadlines for completing an activity (relative to another activity). Reasoning on time intervals is used to verify whether a constraint is violated.

For verifying that a process with loops satisfies a general time-related constraint, typically temporal model checking techniques are applied. The properties of interest are *metric* temporal constraints, e.g., deadline on execution of activities in a business process. *Metric temporal logic* (MTL), a temporal logic with metric temporal constraints, can express typical compliance requirements as presented in this paper. Unfortunately, the model checking problem for MTL is undecidable over models with infinite traces [26]. By introducing so called observers on atomic propositions, the problem whether a process model, given as a timed transition system (TTS), satisfies an MTL formula becomes decidable by a reduction to LTL modelchecking [6]. This approach allows to check temporal compliance of a real-time extension of Dwyer's specification patterns [15]. A similar approach is followed in [19] for checking whether an extended CCSL (Clock Constraint Specification Language) specification holds in a timed Petri net; CCSL is less expressive than the constraints that can be expressed and checked with our technique.

An alternative approach to describe temporal constraints is *timed Declare* [50] in which LTL-like constraints are extended with the notion of time. By a translation to timed automata, such constraints can be monitored at runtime to evaluate whether a process instance might or will violate a temporal constraint. A similar approach is proposed in [35].

In comparison, the technique presented in this report focuses on backwards checking of temporal constraints in execution logs. The generic Petri net pattern proposed in Sect. 9is capable to express all temporal constraints that we encountered in the works discussed above, and other temporal constraints such as cyclic temporal constraints not discussed elsewhere. Our technique detects *all* temporal violations in a trace, not just the first temporal violation encountered as it happens in model checking approaches. In case of violations also the compliant behavior (when a non-compliant event should have happened) is returned as diagnostic information.

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 Preliminaries

This section recalls basic conformance checking notions [2, 9, 29] on which we build for temporal compliance checking.

3.1 Control-Flow Alignment

Conformance checking relates behavior that *has happened* and was recorded in an event log L to a formal specification S that describes which behavior *should have happened*. In this context, an *event log* is a *multiset* of *traces*. Each trace describes a particular *case* (i.e., a *process instance*) as a sequence of *events*. An event often refers to an *activity* executed.

Let E be the finite set of all events and L be an *event log* (a *multiset* of all *traces*). $\sigma_L \in L$ is a trace in the event log L.

Let A be the finite set of all activities and A^* be the set of all sequences of activities over elements in A. We define $S \subseteq A^*$ as the set of all sequences in A^* which are compliant with a certain compliance rule. $\sigma_S \in S$ is a sequence; $\sigma_S = \langle a_1, ..., a_n \rangle$ where $a_i \in A$. Labeling function ℓ relates every activity $a \in A$ to a set of events. i.e., $\ell(a) \subseteq E$.

The set S can be described by a Petri net N such that S is the set of all terminating runs of N. A trace $\sigma_L \in L$ may deviate from S, i.e., $\sigma_L \notin S$. To understand where and how σ_L deviates from S, we use the control-flow alignment approach in [2, 9]. An optimal alignment of σ_L to specification S is a compliant trace $\sigma_S \in S$ that is as similar to σ_L as possible.

A given trace σ_L will be related to trace $\sigma_S \in S$ by pairing events in σ_L to events in σ_S. Formally a *move* of (σ_L and S) is a pair $(x, y) \in (E \cup \{\gg\}) \times (A \cup \{\gg\}) \setminus \{(\gg, \gg)\}$. For $x \in E$ and $y \in A$, 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 $\sigma_L \in L$ to S is a sequence $\gamma = \langle (x_1, y_1), \ldots, (x_n, y_n) \rangle$ of moves (of σ_L and S) such that the projection $x_1, \ldots, x_n$ to E is the original trace σ_L; $\langle x_1, \ldots, x_n \rangle|_E = \sigma_L$, and the projection $y_1 \ldots y_n$ to A is the specified trace σ_S; $\langle y_1, \ldots y_n, \rangle|_A = \sigma_S \in S$.

For example, for specification $S = \{\langle a, b, c, d\rangle, \langle a, c, b, d\rangle\}$ with $\ell(y) = \{y\}$ 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 activities 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 minimized the number of non-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 [9] finds such an optimal alignment using a cost-based approach: a cost-function κ assigns each move (x, y) a cost $\kappa(x, y)$ s.t. a synchronous move has cost 0 and all other types of moves have cost > 0. The A*-based search on the space of (all prefixes of) all alignments of σ_L to S described in [9] can be used to find an optimal alignment for σ_L and S.

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

3.2 Data-Aware Petri-nets

In a process model each process instance is characterized by its case attributes. Different paths of a model may be taken during the execution of a process. These may be governed by guards and conditions defined over such attributes. Process models also define, for each attribute, its domain, i.e., the values that can be given. Moreover process models also describe which attributes every activity can read or write. Such activity attributes can prescribe a resource or group of resources allowed to execute a certain activity, an information object or a collection of information objects the activity has access to or time stamp of the activity execution.

Execution of a process can be fully conforming to its model if (1) the sequence of the activities executed can be replayed on a process path in the model and (2) the respective process instance attributes and activities' attributes also match the specified attributes in the model.

Suppose a Petri-net model N shown in Fig. 1. Petri-net N specifies that firing transition a *writes* a value for *attribute* X and both transitions b and c *read* the value of *attribute* X and *write* the value of *attribute* Y. Petri-net N restricts transition b to fire only if the *written value* of Y is greater than *read value* of X by assigning the guard $[Y' > X]$ to transition b. Likewise it restricts transition c to fire only if the *written value* of Y is equal or smaller than *read value* of X by assigning the guard $[Y' \leq X]$ to transition c.

As explained in Sect. 3.1 an event log is a multiset of traces. An event has a name and refers to a case. A log may also store additional properties of an event such as

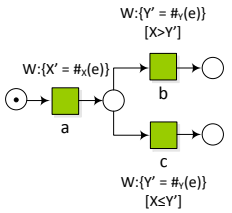

Fig. 1. Data-Aware Petri-net N

a *time stamp* (e.g., the time a certain medicine administered for a patient), a *resource* (e.g., the nurse or doctor executing or initiating the activity), and various *data elements* recorded with the event (e.g., the dosage of the medicine administered).

Consider two traces σ_{L_1} and σ_{L_2} having events with additional attributes: $\sigma_{L_1} = \langle (a, X = 3), (b, Y = 2) \rangle$ and $\sigma_{L_2} = \langle (a, X = 3), (b, Y = 5) \rangle$. The sequence of events in both traces σ_{L_1} and σ_{L_2} conforms with the control flow specified in Petri-net N. However, trace σ_{L_1} does not fully conform because the guard governing the occurrence of event b evaluates to *false* for trace σ_{L_2}. In the following section we describe formally how we apply data-aware Petri-nets for temporal compliance checking.

3.3 Data-Aware Alignment

Let us now assume that each event has a time stamp and can be described by a pair (e, t) where e is the event name and t is the time stamp. A trace $\sigma_L = \langle (e_1, t_1), \ldots, (e_n, t_n) \rangle$ is a non-descending sequence of timed events, i.e., $t_i \geq t_{i+1}$ for $1 \leq i < n$. Similarly, every activity a in the specified trace σ_S is described as a pair (a, T) where T is the set of all times at which activity a is allowed to be executed according to specification S.

A trace $\sigma_L \in L$ and $\sigma_L = \langle (e_1, t_1), , \ldots, (e_n, t_n) \rangle$ *complies* with specification S if there exists a trace $\sigma_S \in S$ s.t. $\sigma_S = \langle (a_1, T_1), ..., (a_n, T_n) \rangle$ where for each $e_i \in E$, there exists an activity $a_i \in A$ where $e_i \in \ell(a_i)$ and $t_i \in T_i$.

For example, assume that specification S includes two admissible traces σ_{S_1} and σ_{S_2}; $S = \{\sigma_{S_1}, \sigma_{S_2}\}$ with $\sigma_{S_1} = \langle (a_1, [3, 3]), (a_2, [4, 4]), (a_3, [13, 14]) \rangle$ and $\sigma_{S_2} = \langle (a_1, [7, 7]), (a_2, [9, 9]), (a_3, [17, 19]) \rangle$. Trace $\sigma_{L_1} = \langle (a_1, 3), (a_2, 4), (a_3, 13.06) \rangle$ complies with specification S (assuming $\ell(y) = \{y\}$) because there is a trace $\sigma_{S_1} \in S$ where the time stamps recorded for the events in σ_{L_1} fall in the admissible time stamps of the same events in σ_{S_1}. Contrary, trace $\sigma_{L_2} = \langle (a_1, 3), (a_2, 4), (a_3, 14.06) \rangle$ does not comply with specification S. Although the sequence of events in σ_{L_2} complies with S, the third time stamp recorded in σ_{L_2} is not admissible based on specification S.

As is mentioned in Sect. 3.1, set S can be described by a Petri net N such that S is the set of all terminating runs of N. As we extend specification S with admissible time stamps of its activities, N is also extended by the respective temporal constraints. Hence extension of the optimal control-flow alignment with recorded temporal information will give an extended alignment in which moves also consider the temporal dimension as follows:

- $((x, t), \gg)$, *move on log*
- $(\gg, (y, T))$, *move on specification* S
- $((x, t), (y, T))$ with $t \notin T$ and $x \in \ell(y)$, *synchronous move with a time-related deviation*
- $((x, t), (y, T))$ with $t \in T$ and $x \in \ell(y)$, *synchronous move with correct time stamp*

The data flow alignment technique in [29] assigns a 'cost$>$ 0' not only for *move on log* and *move on specification* but also for *synchronous moves with a time-related deviation*.

As is explained earlier, the approach we are using for temporal compliance checking is based on the principle of finding an alignment between an event log and a process model. The events in the trace are mapped to the activities in the process model. In the first step the events in the log are replayed over the activities in the model to get control-flow alignment(s) which are closest to the actual trace and have the least control-flow deviation. In the second step we use the control-flow alignment(s) to get the data flow alignment considering time as a data attribute.

To this end the time stamps of events in the control-flow alignment are compared with admissible time stamps in the model to get a data flow alignment which is closest to the time stamps of events in the actual trace and has the smallest temporal deviation.

The A*-based search on the space of (all prefixes of) all alignments of σ_L to S described in [2, 9] can be used to find alignment(s) for σ_L and S which have the least cost of deviation (when attributes of activities are ignored). This approach is extended in [29] to find *data-aware alignments*; an ILP solver finds among all synchronous moves, values for the specified attribute in S such that the data deviations are minimized. In the following, we apply alignments for temporal compliance checking. This two stage approach helps us to create an optimal alignment considering both the control-flow perspective and the time perspective.

4 Compliance Rule Framework

A compliance requirement 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, 17, 10, 15, 23, 47, 21, 45, 46, 28, 27, 38, 6, 50, 30], collected the requirements described in these papers, and categorized them. We found that a single requirement 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 requirement "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.").

Furthermore, a compliance requirement 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. 2. In this report, we present compliance rules for control flow and process time, where we focus on untimed, observation-based properties of individual cases.

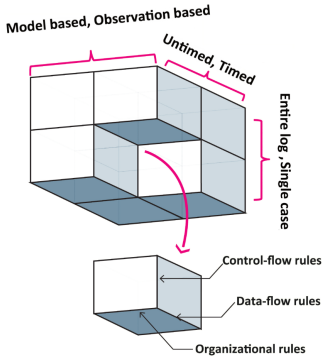

Fig. 2. Compliance Rule Framework

Typically every compliance requirement restricting the process time implies a control flow rule as well. Even if the sequence of execution of activities are not restricted in the compliance requirement, the existence of specified activities must be checked. Only then we can check the temporal constraint governing the execution of these activities. For example assume the compliance requirement "Activity A must be executed at time t"; this requirement is decomposed to two compliance rules; (1) (control flow) Activity A must be executed (2) (process time) Activity A may only be executed at time t. To check the temporal constraint of this compliance requirement we can apply control flow alignment to check if the specified activity is executed or not and then using data-aware alignment, we check if it was executed at the admissible time stamp.

Sect. 5, presents an overview of control flow and temporal compliance rule categories and Sect. 7 and Sect. 10 present all compliance rules and their formalization in form of Petri-net patterns.

5 Collection of Control-Flow and Temporal 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 [42]. 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, 17, 10, 15, 23, 47, 21, 45, 46, 28, 27, 38, 6, 50, 30], 54 control-flow and 15 temporal compliance rules and classified them further respectively into 10 and 7 categories; see Tab. 1 for control-flow compliance rules and Tab. 2 for temporal compliance rules. Each category includes several compliance rules. For example, the *Existence* category from Tab. 1 defines two rules in total: "In each process execution, activity A should be executed" and "In each process execution, activity A

Table 1. Categorization of the 54 Control Flow Compliance Rules

Category (Rules)	**Description**
Existence (2)	Limits occurrence or absence of an activity. [4],[17],[10], [15],[23],[47],[45]
Bounded Existence (6)	Limits the number of times an activity must or must not occur. [17],[15]
Dependent Existence (6)	Limits the presence or absence of an activity with respect to existence or absence of another activity.[17]
Bounded Sequence (3)	Limits the number of times a sequence of activities must or must not occur. [17],[15]
Parallel (2)	Limits occurrence of a specific set of activities in parallel. [45]
Precedence (10)	Limits occurrence of an activity in precedence over another activity. [17],[45],[15],[47],[10],[21],[23],[4],[45]
Chain Precedence (4)	Limits occurrence of a sequence of activities in precedence over another sequence of activities. [17],[15],[23]
Response (10)	Limits occurrence of an activity in response to another activity. [45],[15],[23],[17],[48],[10],[21]
Chain Response (4)	Limits occurrence of a sequence of activities in response to another sequence of activities. [17]
Between (7)	Limits occurrence of an activity within (between) a sequence of activities. [15]

Table 2. Categorization of the 15 Temporal Compliance Rules

Category (Rules)	**Description**
Instance Duration (2)	Limits the time length in which a control-flow rule instance must hold. [50]
Delay Between Instances (1)	Limits the delay between two subsequent instances of a control-flow rule [28, 27, 38, 6]
Validity (3)	Limits the time length in which an activity can be executed.[28, 27, 38, 50]
Time Restricted Existence (2)	Limits the execution time of an activity in calendar.[28, 27, 38]
Repetition (2)	Limits the delay between execution of two subsequent activities.[28, 27, 38, 50, 30, 6]
Time Dependent variability(1)	Limits choice of a process path among several ones with respect to temporal aspects.[28, 27, 38, 50]
Overlap (4)	Limits start and completion of an activity to start and completion of another activity.[28, 27, 38, 50]

should not be executed." Each rule is parameterized over activities (e.g., Activity A) or numeric parameters (e.g., governing bounds for repetitions etc.).

In Sect. 7 we will present the complete collection of control-flow compliance rules and their formalization. The complete collection of temporal compliance rules and their formalization are described in Sect. 10.

6 Control-Flow Compliance Checking

As briefly mentioned in the previous section every compliance rule from our collection of control-flow compliance rules is formalized in terms of *parameterized Petri-net pattern*. Suppose a compliance requirement stating: "The treatment with antibiotics must be administered with one dose per day for 3 days in a row. After each cycle of treatment, in case of necessity, the treatment can be extended for other cycles; but there should be delay of at least one week between two subsequent cycles of treatment". This requirement constraints both process time and control-flow perspective of process and it can be divided into three different compliance rules: (1) (control flow)"antibiotics must be administered in cycles of 3 occurrences", (2) (process time) "between two subsequent administration of antibiotics in a cycle, there should be one day delay", and (3) (process time) "between two subsequent cycles, there should be at least one week delay".

The Petri-net pattern formalizing the control-flow compliance rule in this example (for $k = 3$) is illustrated in Fig. 3.

The core of the rule is formalized in the grey-shaded part between transitions I_{st} and I_{cmp}. The rule becomes $active$ when I_{st} occurs. Then activity A has to occur 3 times before the rule can complete (each time A occurs, one token is taken from $todo$ and put on $done$). In between, arbitrary other activities can occur, expressed by transition Ω. The compliance rule may hold multiple times in a trace; this behavior is captured by the cycle involving I_{st} and I_{cmp}. Whenever I_{st} occurs, it puts 3 tokens in the place $todo$ which activates a new *instance* of the rule "activity A occurs in groups of 3"; the instance completes with transition I_{cmp} which removes all tokens from $done$ and puts a token on p_1.

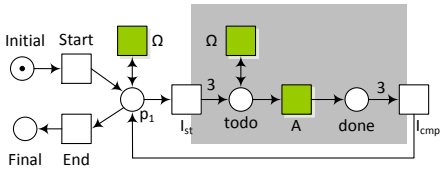

Fig. 3. Petri net formalizing a control-flow rule.

We need to distinguish different *instances* of a rule from each other. Hence, every compliance pattern has a *Pattern Instance* starts by the occurrence of an activity which activates the compliance rule and ends as soon as the compliance rule is satisfied. In our running example "3 occurrences of A in a cycle".

The entire Petri net of Fig. 3 thus allows for multiple instances of the compliance rule, each instance is framed by the I_{st} and I_{cmp} transitions; between two instances arbitrary other activities are allowed as expressed by the Ω-transition attached to place p_1. Each activity A of the compliance rule is represented in the Petri-net pattern as a transition with label A. In general there could be multiple transitions with label A. In case of our example, activity A = *antibiotic administration*. Occurrences of other activities than the activity(s) specified in the compliance rule are described by the Ω-labeled transitions. This way, the pattern abstracts from all other trace activities that are not described in the compliance rule.

The net has a dedicated place *Initial* and a place *Final*, a *Start* and an *End* transition. A compliant behavior takes the net from the initial marking to the final marking (just one token in $Final$) showing arbitrary many instances of the compliance rule.

The alignments of Sect. 3.1 can be used to check compliance of a trace to the compliance rule "activity A must be executed in groups of k occurrences". Assume the trace

$\sigma = \langle (B,1)(A,2)(A,30)(A,54)(A,100)(C,123)(A,162)(D,173) \rangle$ to be given. The first element of each pair in the sequence σ represents the *activity*, the occurred event is refereing to and the second element records the value for the time attribute of corresponding activity. That is the time value every activity was executed. (Please note that, the time value recorded for occurrence of each activity has no explicit notion of time; the time values simply recorded numerical values of the event attribute time.)

For control-flow compliance checking, we ignore attributes of the events and thus align the trace $\sigma = \langle BAAAACAD \rangle$ to the net of Fig. 3. When aligning σ, A maps to A and Ω maps to all other events B, C, D which are not relevant for the rule. Additionally, we assume transitions $Start$, End, I_{st} and I_{cmp} to be *silent* so that moves on the specification (without corresponding event in σ) have cost 0. The approach of Sect. 3 yields a best alignment $\gamma_1 = \frac{\gg |B| \gg |A|A|A| \gg |\gg |A|C| \gg |A| \gg |D| \gg}{Start|\Omega|I_{st}|A|A|A|I_{cmp}|I_{st}|A|\Omega|A|A|I_{cmp}|\Omega|End}$ showing 1 instance of the rule with 3 occurrences of A and 1 instance of the rule with 2 occurrences of A. The alignment also shows a missing event A in the second instance by the move on specification $(\gg, A)$. Note that σ has 6 more best alignments to the net of Fig. 3 varying in where the move $(\gg, A)$ is placed. The subsequent steps (shown for γ_1) would have to be executed for each of these alignments.

7 Collection of Control Flow Compliance Rules and Their Formalization

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*.
- 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).
- Every compliance pattern has a *Pattern Instance* corresponding to an instance of its compliance rule. The *Pattern Instance* starts as soon as an event occurs which triggers the *Compliance Rule Instance*. The *Pattern Instance* completes as soon as the condition of the *Compliance Rule Instance* is satisfied.
- The I_{st}-labeled transition in every Petri-net pattern indicates the start of an instance of a control flow pattern (*Pattern Instance*) and the I_{cmp}-labeled transition in every pattern indicates the completion of an instance of the same control flow pattern.
- Σ_L denotes the set of activity names for compliance rules. Depending on the choice of compliance rules, it may include elements describing start and completion of activities.
- 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 activities A, B, and C; hence the events A, B, and C in trace 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 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 End is enabled, and its occurrence leads to the *final marking*.
- $Start$-labeled, End-labeled, I_{st}-labeled, I_{cmp}-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 3.1, assigns the cost of zero for deviation of these 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 transition . This arc ensures that when the corresponding transition fires all tokens are consumed from the respective place (even if it contains no token). We usually use *reset arcs* to consume all tokens from the net, thereby guaranteeing that after firing End, no transition is enabled anymore and the net is empty.
- In some patterns we connect a place to a transition with an *inhibitor arc*. An *inhibitor arc* ensures that corresponding transition can only fire if the place at the other end of the *inhibitor arc* is empty.

7.1 Existence Category

This category contains compliance rules which limit the occurrence or absence of an activity within a chosen scope[1].

Existence. Activity universality .
Description: Activity A must occur within a chosen scope. The compliance rule is violated if event A does not occur within the specified scope (e.g., a process instance). An

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

instance of this compliance rule includes one time occurrence of A i.e., if A occurs once this compliance rule is satisfied. Figure 4 shows the Petri-net pattern that formalizes this rule.

After the pattern started, any event may occur. The *pattern instance* is trigged as soon as A is executed. With the execution of A, the corresponding compliance rule of this pattern is satisfied and the *pattern instance* is completed. After completion of the *pattern instance*, any event may occur. In this situation the pattern may terminate. The transition End models that the end of the trace has been reached i.e., it occurs *after* all events of the trace occurred.

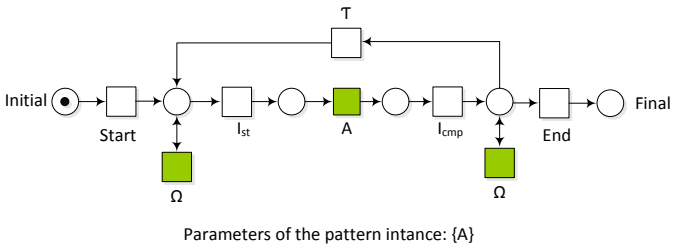

Fig. 4. 'Existence. Activity universality' Compliance rule

Existence. Activity absence .
Description: Activity A must not occur within a chosen scope. The rule is violated if activity A occurs within the specified scope. An instance of this compliance rule includes occurrence of any activity except A. That is, as long as A does not occur, this compliance rule is satisfied but if A occurs this rule is violated. Figure 5 shows the Petri-net pattern that formalizes this rule.

The pattern specifies that any event but A may occur. Therefore by construction in every instance of this pattern only the Ω-labeled transition is enabled. If an A occurs a deviation is captured. The transition End models that the end of the trace has been reached i.e., it occurs *after* all events of the trace occurred.

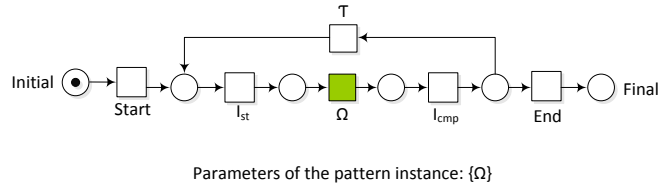

Fig. 5. 'Existence. Activity absence' Compliance rule

7.2 Dependent Existence Category

This category of compliance rules limits the presence or absence of an activity with respect to existence or absence of another activity.

Dependent Existence. Exclusive .
Description: Presence of activity A mandates the absence of activity B within a chosen scope. This rule is violated if within the specified scope, both activities A and B would be present. An instance of this compliance rule includes all occurrences of activity A or all occurrences of activity B. The Petri-net pattern illustrated in Figure 6 formalizes this rule.

After the pattern started, any activity may occur. The *pattern instance* is activated as soon as the first A or B occurs. If 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 by firing one of the transitions End.

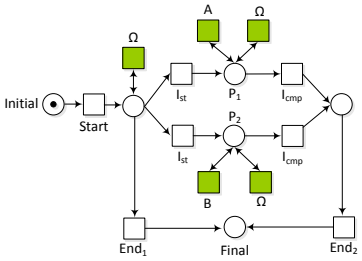

Fig. 6. 'Dependent Existence. Exclusive' Compliance rule

Dependent Existence. Mutual exclusive Description: Within a chosen scope, either activity A or activity 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 within the specified scope. An instance of this compliance rule includes all occurrences of activity A or all occurrences of activity B. The Petri-net pattern illustrated in Figure 7 formalizes this rule.

The behavior of this pattern is similar to the pattern described in Figure 6; with the difference that in the pattern illustrated in Figure 7 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 unless the *pattern instance* is completed i.e., only after the occurrence of one of the events A or B. After the condition of the rule is satisfied, the pattern may terminate by firing transition End.

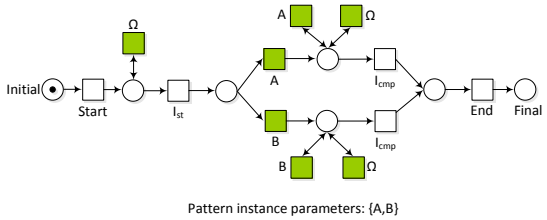

Fig. 7. 'Dependent Existence. Mutual exclusive' Compliance rule

Dependent Existence. Prerequisite Description: Absence of activity A mandates that activity B is also absent within a chosen scope. This rule is violated if within the specified scope, activity B occurs without any occurrence of activity A. This compliance category consists of one compliance rule. An instance of this compliance rule includes occurrences of both activities A and B. The Petri-net pattern illustrated in Figure 8 formalizes this rule.

Occurrence of any activity triggers the *pattern instance*. 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 instance* can complete and 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 occurrence of B. Please note that sequence of occurrences of A and B does not matter but if B occurs, A must have occurred at least once before it or it must eventually occurs after it. 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. Please note that absence of both activities is allowed based on the rule. When the condition of the rule is satisfied, the *pattern instance* is completed and the pattern may terminate by firing the transition End.

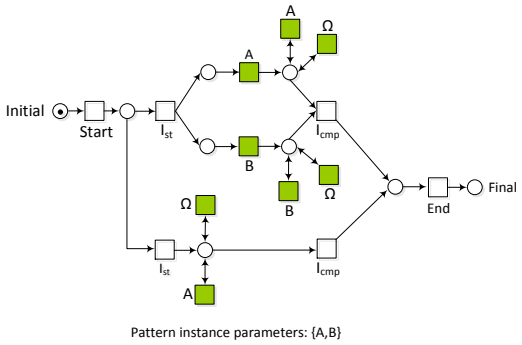

Fig. 8. 'Dependent Existence. Prerequisite' Compliance rule

Dependent Existence. Inclusive Description: Presence of activity A mandates that activity B is also present within a chosen scope. This rule is violated if within the specified scope, activity A occurs without any occurrence of activity B. This compliance category consists of one compliance rule. An instance of this compliance rule includes all occurrences of activities A and B. The Petri-net pattern illustrated in Figure 9 formalizes this rule.

The pattern described in Figure 9 is similar to the pattern described in Figure 8; with the difference in adjacent transitions to the place P in Figure 9.

Occurrence of any activity triggers the *pattern instance*. 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. When the condition of the rule is satisfied, the *pattern instance* completes. This is the situation that the pattern can terminate by firing the transition End, if no activity is to be executed. Please note that it is possible that B occurs without occurrence of A or none of A or B occurs.

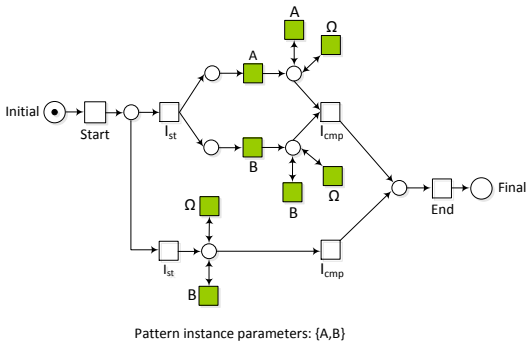

Fig. 9. 'Dependent Existence. Inclusive' Compliance rule

Dependent Existence. Substitute Description: Activity B substitutes the absence of activity A within a chosen scope. This rule is basically the logical OR between occurrences of two activities A and B. This rule is violated if within the specified scope, non of the activities A or B occurs (i.e., both be absent). This compliance category consists of one compliance rule. An instance of this compliance rule includes all occurrences of activities A and B. The Petri-net pattern illustrated in Figure 10 formalizes this rule.

Occurrence of any activity triggers the *pattern instance*. The *pattern instance* cannot complete unless at least one of the activities A or B occur. The occurrence of A does not oblige the occurrence or non-occurrence of B, however its absence obliges the occurrence of B. When the condition of the rule is satisfied, the pattern may terminate by firing the transition End.

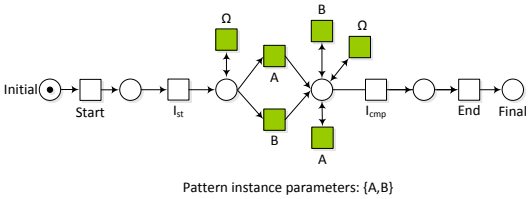

Fig. 10. 'Dependent Existence. Substitute' Compliance rule

Dependent Existence. Co-requisite Description: Within a chosen scope, either activities A and B should exist together or be absent together. This rule is violated if within the specified scope, only one of the activities A or B occurs. This compliance category consists of one compliance rule. An instance of this compliance rule includes all occurrences of A and B if they occur. The Petri-net pattern illustrated in Figure 11 formalizes this rule.

The pattern described in Figure 11 is similar to the pattern described in Figure 8; with the difference in adjacent transitions to the place P.

Occurrence of any activity triggers the *pattern instance*. As soon as activity 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 activity 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. Please note that absence of both activities A and B is also compliant. When the condition of the rule is satisfied, the *pattern instance* completes and the pattern may terminate by firing the transition End.

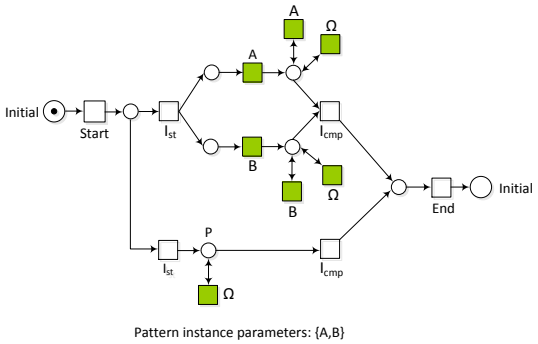

Fig. 11. 'Dependent Existence. Co-requisite' Compliance rule

7.3 Bounded Existence Category

This category includes compliance rules that limit the number of times an activity must or must not occur.

Bounded Existence of an Activity. Exactly *k* times .
Description: Activity A must occur exactly k times within a chosen scope. The rule is violated if A occurs less than or more than k times within the specified scope. An instance of this compliance rule includes k occurrences of activity A. That is, as soon as A occurs k times, this rule is satisfied. Figure 12 shows the Petri-net pattern that formalizes this rule for the case $k = 2$.

After the pattern started any event may occur. The first occurrence of A triggers the *pattern instance*. The pre-place P_k of A is initially marked with k tokens. The k tokens in place P_k assure that activity 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 compliance rule is satisfied and the *pattern instance* is completed. The pattern can terminate only if the *pattern instance* is completed implying that the condition of the rule is satisfied. The transition End models that the end of the trace

has been reached. Please note that Ω-labeled transition is enabled throughout the whole pattern and may occur at any point in time.

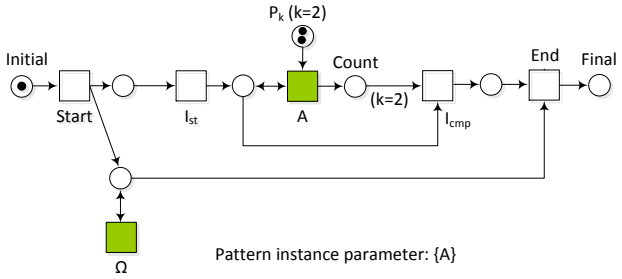

Fig. 12. 'Bounded Existence of an Activity. Exactly k times' Compliance rule

Bounded Existence of an Activity. At least k times .
Description: Activity A must occur at least k times within a chosen scope. If A occurs less than k times within the specified scope, the rule is violated. An instance of this compliance rule includes k occurrences of activity A. That is, as soon as A occurs k times, this rule is satisfied. In addition, the rule also allows for more occurrences of A. Figure 13 shows the Petri-net pattern that formalizes this rule for the case $k = 2$.

The basic structure of the pattern in Figure 13 is similar to the pattern described in Figure 12. Occurrence of the first A activates the *pattern instance*. 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, hence the *pattern instance* is completed. The rule specifies that A must occur at least k times; therefore after k occurrences of A, further occurrences of A are possible. The Ω-labeled transition is always enabled. The transition End models that the end of the trace has been reached.

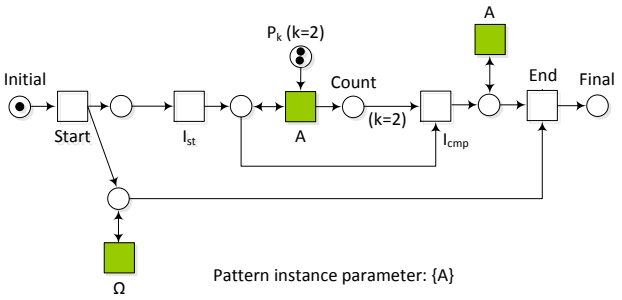

Fig. 13. 'Bounded Existence of an Activity. At least k times' Compliance rule

Bounded Existence of an Activity. At most k times .
Description: Activity A must occur at most k times within a chosen scope. If A occurs more than k times within the specified scope, the rule is violated. An instance of this compliance rule includes at most k occurrences of activity A. That is, if A occurs

less than $k + 1$ times, this rule is satisfied. Figure 14 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 showed in Figure 12. However in contrast with the pattern described in Figure 12, this rule allows for less than $k + 1$ occurrences of A. Occurrence of any activity activates the *pattern instance* of this rule even if it is not an A; because the condition of this rule is satisfied even if A does not occur at all. The pre-place 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 *pattern instance* is completed when there is no activity to happen even if there are tokens left in place P_k (k, less than k or zero tokens). When the *pattern instance* completes, it removes all tokens left in place P_k by the *reset arc* connecting place P_k to I_{cmp} (represented as a two arrow headed line). The transition End models that the end of the trace has been reached.

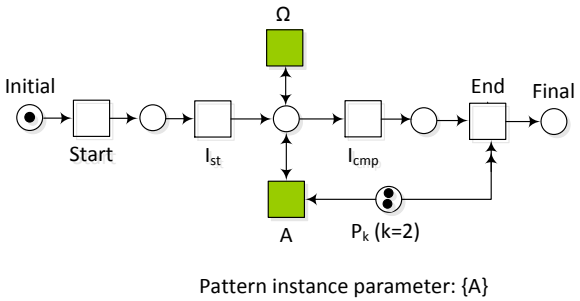

Fig. 14. 'Bounded Existence of an Activity. At most k times' Compliance rule

Bounded Existence of an Activity. Exactly *k* times in a row .
Description: Activity A must occur exactly k times in a row (directly one after the other) within a chosen scope. The rule is violated if the sequence $\langle \underbrace{A, \ldots, A}_{k} \rangle$ does not occur within the specified scope. An instance of this compliance rule includes k occurrences of activity A in a row. That is, if A occurs exactly k times without any other activities occurring between occurrences of A, this rule is satisfied. Figure 15 shows the Petri-net pattern that formalizes this rule for the case $k = 2$.

After the pattern started, any event may occur. Occurrence of the first A activates the *pattern instance*. After the start of the *pattern instance*, no Ω-labeled transition is enabled anymore until A occurs k times in a row. The k tokens in pre-place P_k of A limits the occurrences of A to k times (similar to the structure described in Figure 12). The place $Count$ counts the number of occurrences of A. The *pattern instance* completes only if there are k tokens in place $Count$, implying that the condition of the rule is satisfied. In this situation the *pattern instance* completes and any non-A event may occur (captured by the Ω-labeled transition) or the pattern may terminate by firing the transition End.

Bounded Existence of an Activity. At least *k* times in a row .

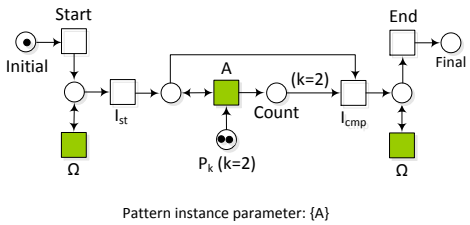

Fig. 15. 'Bounded Existence of an Activity. Exactly k times in a row' Compliance rule

Description: Activity A must occur at least k times in a row (directly one after the other) within a chosen scope. This rule is violated if $\langle \underbrace{A, \ldots, A}_{k} \rangle$ does not occur within the chosen scope. An instance of this compliance rule includes k occurrences of activity A in a row. That is, as soon as A occurs exactly k times without any other activities occurring between occurrences of A, this rule is satisfied. Figure 16 shows the Petri-net pattern that formalizes this rule for the case $k = 2$.

After the pattern started, any activity may occur including A. As soon as the first activity A occurs, we distinguish two scenarios: $Scenario1$ is that $(k)A$'s occur in a row; $Scenario2$ is that an Ω occurs after less than $(k)A$'s in a row.

In case of $Scenario1$, as soon as the first A occurs the *pattern instance* is activated (please see the shadowed subnet labeled (b)) and no Ω-labeled transition is enabled any more until the *pattern instance* completes. The k tokens in the place P_k limits the number of occurrences of A to k in the *pattern instance*. The *pattern instance* completes only if the sequence $\langle \underbrace{A, \ldots, A}_{k} \rangle$ occurs; implying that the condition of the rule is satisfied. The *inhibitor* arcs connecting the place P_k to the transition I_{cmp} assures that the I_{cmp} can fire only if the place P_k is empty. The compliance rule specifies that more than k occurrences of A in a row is allowed; hence after the compliance rule is satisfied any arbitrary occurrences of A or Ω is possible. In this situation the pattern may terminate by firing the transition End.

The left part of the pattern (please see the shadowed subnet labeled (a)) models the $Scenario2$. That is, A may occur $k-1$ times in a row. The $k-1$ tokens in the place P_{k-1} limits the occurrences of A directly one after the other to $k-1$ in the subnet (a). Please note that A may occur arbitrary number of times in the subnet (a) as long as, the sequence of A's is interrupted by an Ω-labeled activity before the sequence $\langle \underbrace{A, \ldots, A}_{k} \rangle$ occurs. The pattern may only terminate if the condition of the rule is satisfied implying that the *pattern instance* must be executed.

Bounded Existence of an Activity. At most *k* times in a row .
Description: Activity A must not occur more than k times in a row (directly one after the other) within a chosen scope. This rule is violated if $\langle \underbrace{A, \ldots, A}_{>k} \rangle$ occurs within the specified scope. An instance of this compliance rule includes all occurrences of activity A. Figure 17 shows the Petri-net pattern that formalizes this rule for the case $k = 2$.

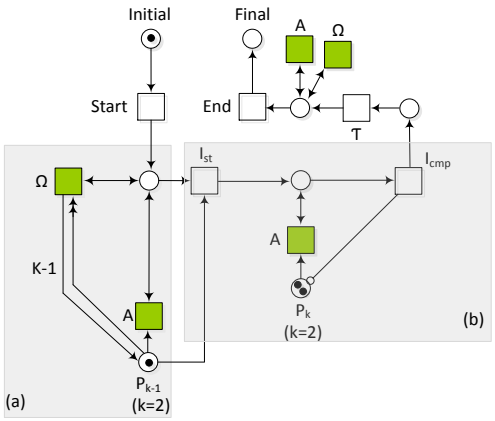

Fig. 16. 'Bounded Existence of an Activity. At least k times in a row' Compliance rule

After the pattern started, the place P is marked. At this marking any activity may occur. Based on this rule, less than k occurrences of A or exactly k occurrences of A in a row are considered as compliant behavior. Therefore at the marking where the place P is marked (no A has occurred yet), the pattern may terminate by firing the transition End_1.

The *pattern instance* is triggered as soon as the first A occurs. Firing the transition I_{st} produces k tokens in the place P_k; implying that A may occur at most k times in a row in every *instance* of the pattern. Please note that the compliance rule allows for less than k occurrences of A in a row, hence the *pattern instance* can complete even if there are tokens left in the place P_k (less than k or zero tokens). The *reset arc* connecting the place P_k to the transition I_{cmp} removes all the remaining tokens in the place P_k. After the *pattern instance* completes, the pattern may terminate (by firing the transition End_2) because the condition of the rule is satisfied so far. However if an Ω-labeled activity occurs, the pattern returns to the marking where the place P is marked. At this marking an Ω-labeled activity may occur, another *instance* of the pattern may be triggered or the pattern may terminate.

Bounded Existence of an Activity. Bursts of k Occurrences .

Description: Activity A must occur in bursts of k occurrences within a chosen scope. This rule is violated if A does not occur in bursts of k occurrences within the specified scope. An instance of this compliance rule includes k occurrences of A. Figure 18 shows the Petri-net pattern that formalizes this rule.

After the pattern started, any activity may occur. The *pattern instance* is triggered as soon as the first A occurs. Start of the *pattern instance* produces k tokens in place P_k, which restricts the number of occurrences of A in every *pattern instance* to at most k times. Occurrence of each A decrements the number of tokens in P_k and increases the number of tokens in $Count$. The *pattern instance* may complete only if there are k

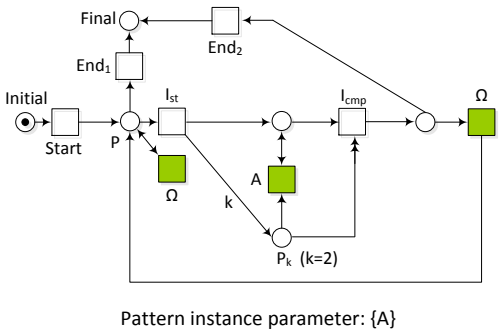

Fig. 17. 'Bounded Existence of an Activity. At most k times in a row' Compliance rule

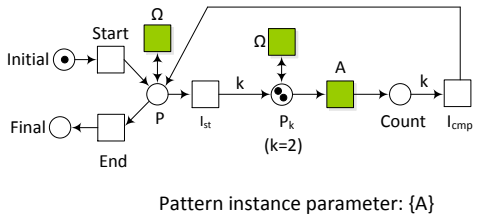

Fig. 18. 'Bounded Existence of an Activity. Bursts of k Occurrences' Compliance rule

tokens in place $Count$. When the *pattern instance* is completed, the pattern returns to the marking where there is a token in place P.

The pattern structure in Figure 18 is modeled s.t. it models a cyclic behavior of occurrences of A. That is, A may occur any arbitrary number as long as it occurs in bursts of k times. Therefore after completion of every instance of the pattern, the next *pattern instance* may start, an Ω-labeled activity may occur or the pattern may terminate (by firing the transition End) because the condition of the rule is satisfied. Please note that every occurrence of A is captured in a *pattern instance* i.e., A may not occur outside of the *pattern instance*.

Bounded Existence of an Activity. n Bursts of k Occurrences .
Description: Activity A must occur in n bursts of k occurrences within a chosen scope. This rule is violated if A does not occur in n bursts of k occurrences within the specified scope. An instance of this compliance rule includes all occurrences of A. Figure 19 shows the Petri-net pattern that formalizes this rule for the case $k = 2$ and $n = 3$.

After the pattern started any activity may occur. The *pattern instance* is triggered as soon as first A occurs. The k tokens in the place P_k assures that in each burst, A occurs exactly k times. Every occurrence of A decrements a token from P_k and increments the number of tokens in place $Count$. Firing the transition τ represents the completion of one burst. Completion of each burst empties the place $Count$; implying that A occurred exactly k times and returns the pattern to the marking where the next burst can be executed by producing k tokens in place P_k. In addition, completion of each burst

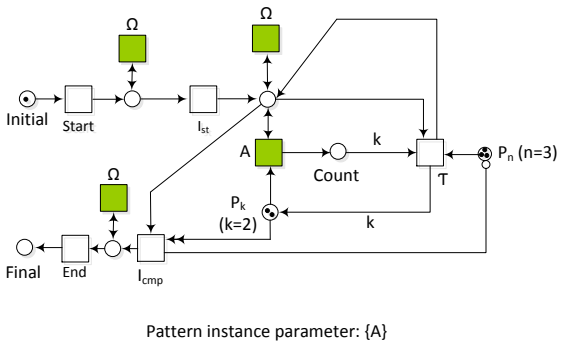

Fig. 19. 'Bounded Existence of an Activity. n Bursts of k Occurrences' Compliance rule

decrements a token from place P_n. The n number of tokens in place P_n limits the execution of bursts to n number.

The *pattern instance* may complete only if the place P_n is empty; implying that n bursts were executed. Please note that Ω-labeled transition is always enabled and it may occur between occurrences of A's. After the completion of the *pattern instance* the pattern may terminate by firing the transition End.

7.4 Bounded Sequence Category

This category includes compliance rules that limit the number of times a sequence of activities must or must not occur within a chosen scope.

Bounded Sequence of Activities. One to one coexistence .
Description: For every activity A, there should exist one activity B and for every activity B there should exist one activity A. If A and B do not occur in form of a pair, the rule is violated. An instance of this compliance rule includes one occurrence of the pair (A, B) in any order. Figure 20 shows the Petri-net pattern that formalizes this rule.

After the pattern started, any activity may occur. Occurrence of first A or B triggers the *pattern instance*. If the pattern starts with an A, B must follow it eventually. Symmetrically occurrence of the first B requires that A follows it eventually, otherwise the *pattern instance* cannot complete. At this situation the pattern may terminate by firing the transition End.

Bounded Sequence of Activities. Coexistence .
Description: For any given number of activities A, there should exist the same number of activities B within a chosen scope. This rule is violated if the number of occurrences of A is not equal to the number of occurrences of B. An instance of this compliance rule includes all occurrences of activities A and B. Figure 21 shows the Petri-net pattern that formalizes this rule.

After the pattern started, any activity may occur. Occurrence of the first A or B triggers the *pattern instance*. In the upper subnet illustrated in the *pattern instance*, place P_1 counts occurrences of A. Each occurrence of activity A increments the number

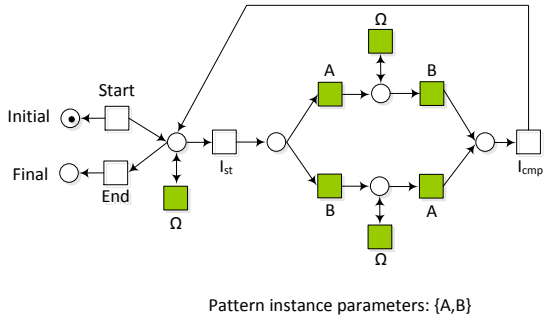

Fig. 20. 'Bounded Sequence of Activities. One to one coexistence' Compliance rule

of tokens in P_1, and each occurrence of activity B decrements the number of tokens in P_1. This construction ensures that for each occurrence of activity 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 instance*, place P_2 counts occurrences of B. Each occurrence of B increments the number of tokens in P_2, and each occurrence of A decrements the number of tokens in P_2. This construction ensures that for each occurrence of activity A, B must have occurred earlier. Therefore place P_2 becomes empty only if A occurs as many times as B has occurred.

The *pattern instance* can complete only if the 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 to the transition I_{cmp} ensure that the *pattern instance* can complete only if P_1 and P_2 are both empty. After the completion of the *pattern instance*, the pattern may terminate by firing the transition End.

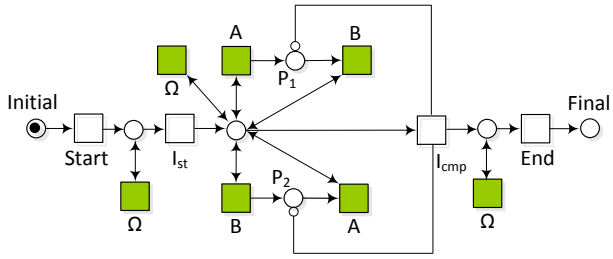

Fig. 21. 'Bounded Sequence of Activities. Coexistence' Compliance rule

Bounded Sequence of Activities. Exactly k times .
Description: The sequence of activities $\langle A_1, ..., A_n \rangle$ must occur exactly k times within a chosen scope. The compliance rule is violated if $\langle A_1, ..., A_n \rangle$ does not occur in the specified sequence or not k times. An instance of this compliance rule includes k occurrences of the sequence $\langle A_1, ..., A_n \rangle$. Figure 20 shows the Petri-net pattern that formalizes this rule for the case $k = 2$.

After the pattern started, any event may occur. The *pattern instance* starts as soon as the first activity in the sequence $\langle A_1, ..., A_n \rangle$ occurs. Occurrence of A_1 decrements a token from the pre-place P_k of A_1. After Occurrence of the first A_1, any event may occur but eventually A_2 must occur. A_1 may be executed only after the first sequence of $\langle A_1, ..., A_n \rangle$ completes. The pattern instance can complete only if the places named P_k are empty, implying that every element of the sequence $\langle A_1, ..., A_n \rangle$ occurred exactly k times. The *inhibitor arcs* connecting the places named P_k to the transition I_{cmp} assure that I_{cmp} may only fire if the places named P_k are empty. After the *pattern instance* completed any activity may occur or the pattern may terminate by firing the transition End.

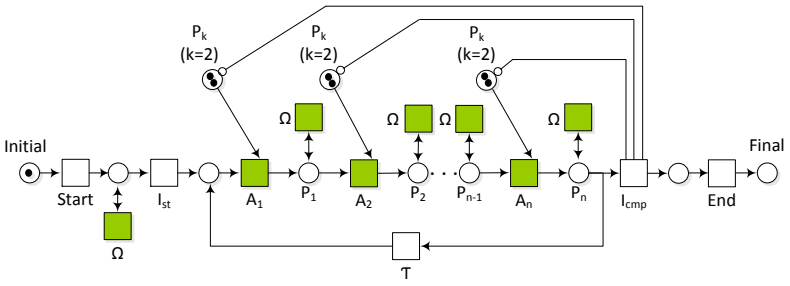

Fig. 22. 'Bounded Sequence of Activities. Exactly k times' Compliance rule

7.5 Parallel Category

This category includes compliance rules that limit the occurrence of an activity in parallel with or during another activity.

Parallel. Simultaneous .
Description: Activity A must always occur in parallel with activity B within a chosen scope. The compliance rule is violated if A and B does not occur simultaneously. The instance of this compliance rule includes start and completion of both activities A and B. Figure 23 shows the Petri-net pattern that formalizes this rule.

As it was mentioned earlier, 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}). After the pattern started, any event may occur. The *pattern instance* is triggered as soon as activity A or B starts.

If A starts, B should also start, i.e., activity A cannot complete unless B has already started. Similarly if start of the activity B activates the *pattern instance*, it is required that activity A also starts. The pattern enforces by construction that the activities A and B must complete together otherwise the *pattern instance* cannot complete, hence the pattern may not terminate. Please note that Ω-labeled activity may occur independently

from occurrences of A and B throughout the pattern. The transition End models that the end of the trace has been reached.

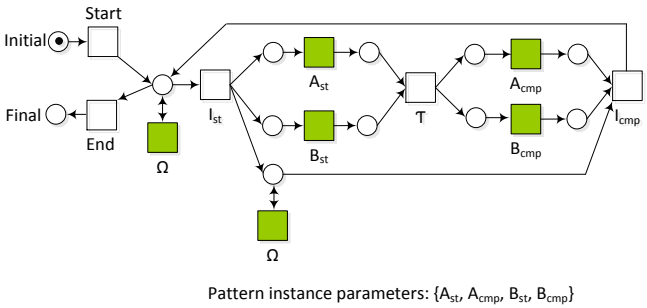

Fig. 23. 'Parallel. Simultaneous' Compliance rule

Parallel. During .
Description: Activity B must be executed during activity A within a chosen scope. The compliance rule is violated if B does not occur within execution of A. The instance of this compliance rule includes start and completion of both activities A and B. Figure 24 shows the Petri-net pattern that formalizes this rule.

After the pattern started, any activity may occur. The *pattern instance* is triggered as soon as activity A starts. Activity B may start only after A has started and B must complete before A completes. The pattern enforces by construction that if B occur, it must be executed during the execution of A. Please note that Ω-labeled activity may occur independently from occurrences of A and B throughout the pattern. The transition End models that the end of the trace has been reached.

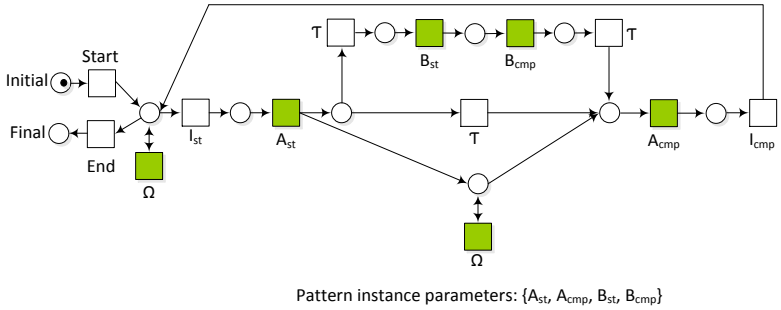

Fig. 24. 'Parallel. During' Compliance rule

7.6 Precedence Category

This category includes compliance rules that limit the occurrence of a one activity in precedence over other activities.

Precedence. Simultaneous or before .
Description: Activity A must always occur before or simultaneously with activity B within a chosen scope. This rule is violated if activity A occurs after activity B within the specified scope. An instance of this compliance rule includes start and completion of both activities A and B. The Petri-net pattern illustrated in Figure 25 formalizes this rule.

Start of activity B or start of activity A which will be followed by B triggers the *pattern instance*. 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 as soon as A starts activity B must be executed as well, i.e., B cannot complete unless A has already started. This is specified by the pre-place P of B_{cmp}. The *pattern instance* may only terminate if both activities A and B complete.

The case where A completes directly before B starts, is also described in the main cycle of the *pattern instance*. After the *pattern instance* started, A starts and completes and directly after that B starts and completes. In this case also completion of both activities A and B is required such that *pattern instance* can complete, hence the pattern may terminate by firing the transition End.

There is no part of the pattern that permits the execution of activity B without a preceding or simultaneous A. If there is no B or if A just occurred, the *pattern instance* is not activated and any event but B_{st} and B_{cmp} may occur. This is also the situation when the pattern may terminate by firing the transition End.

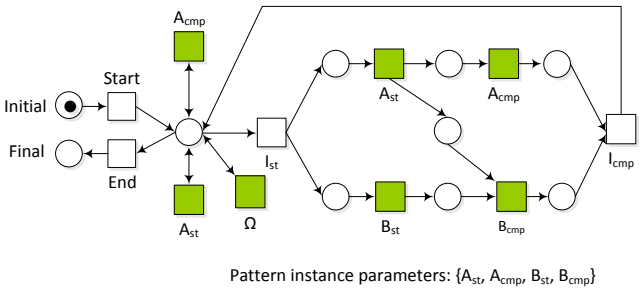

Fig. 25. 'Precedence. Simultaneous or before' Compliance rule

Precedence. Direct .
Description: Every activity B must be preceded by activity A within a chosen scope. If B occurs without a directly preceding A within the specified scope, the rule is violated. An instance of this compliance rule includes execution of activity B and its preceding A. The Petri-net pattern illustrated in Figure 26 formalizes this rule.

The *pattern instance* is triggered as soon as an A occurs which is followed by B. The *pattern instance* describes a cycle of A and B, such that B can only occur if A has directly preceded it. In this situation the *pattern instance* can complete and the pattern may terminate. If there is no B or A just occurred, any activity may occur. This is also the situation when the pattern may terminate by firing the transition End.

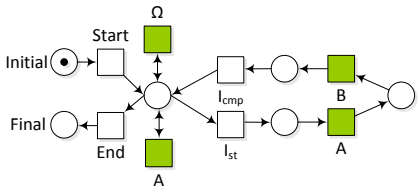

Fig. 26. 'Precedence. Direct' Compliance rule

Precedence. Direct or indirect .
Description: Every activity B must be preceded (directly or indirectly) by activity A within a chosen scope. If A does not occur before B within the specified scope, the rule is violated. An instance of this compliance rule includes execution of activity B and its preceding A. The Petri-net pattern illustrated in Figure 27 formalizes this rule.

The pattern described in Figure 27 is similar to the pattern described in Figure 26; with the difference that the adjacent Ω-labeled transition to place P in Figure 27, allows the indirect precedence of B with A.

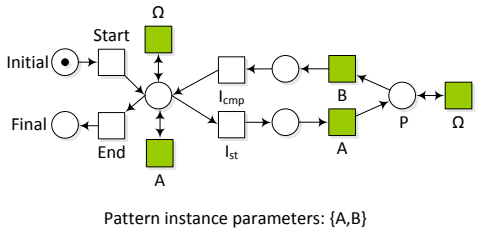

Fig. 27. 'Precedence. Direct or indirect' Compliance rule

Precedence. At least once .
Description: Every activity B must be preceded by activity A at least once within a chosen scope. If A does not precede B at least for one time within the specified scope, the rule is violated. An instance of this compliance rule includes execution of all B activities and a preceding A. The Petri-net pattern illustrated in Figure 28 formalizes this rule.

After the pattern started any activity including A may occur. As soon as an activity A occurs which is followed by first B, the *pattern instance* starts. The *pattern instance* structure describes that B can only occur if before it at least one time A has occurred. As soon as A occurs, place P is marked. In this marking B may occur arbitrary number of times; because the condition of the rule is satisfied. After the *pattern instance* is completed, the pattern may terminate by firing the transition End.

The pattern described in Figure 28 is similar to the pattern described in Figure 27; with the difference that the adjacent B transition to place P in Figure 28, allows arbitrary number of occurrences of B.

Precedence. Direct multiple activities .

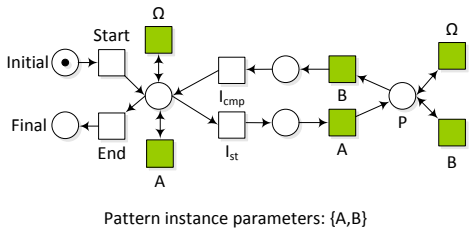

Fig. 28. 'Precedence. At least once' Compliance rule

Description: Every activity B must be preceded directly by another execution of activity B or execution of activity A within a chosen scope. If directly before B one of the activities B or A does not occur within the specified scope, the rule is violated. An instance of this compliance rule includes all occurrences of B and a preceding A. The Petri-net pattern illustrated in Figure 29 formalizes this rule.

After the pattern started any activity including A but B may occur; because before the first B at least once A must have occurred. As soon as an activity A occurs which is followed by the first B, the *pattern instance* starts. After the first occurrence of A, place P is marked. In this marking B can occur an arbitrary number of times and still the condition of the rule is satisfied. That is, the pattern structure describes that B can only occur if directly before it any of the activities A or B was executed. After the *pattern instance* is completed, the pattern may terminate by firing the transition End.

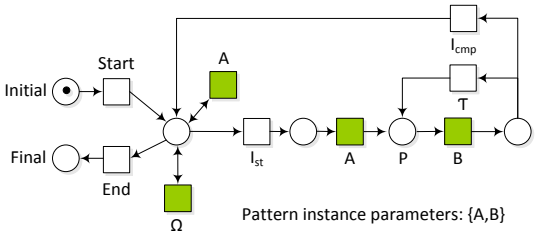

Fig. 29. 'Precedence. Direct multiple events' Compliance rule

Precedence. Direct or indirect multiple activities .
Description: Every activity B must be preceded by another execution of activity B or execution of activity A within a chosen scope. If before B one of the activities B or A does not occur within the specified scope, the rule is violated. An instance of this compliance rule includes all occurrences of B and a preceding A. The Petri-net pattern illustrated in Figure 30 formalizes this rule.

The pattern in Figure 30 is similar to the pattern described in Figure 29; with the difference that as long as B is preceded (even indirectly) by any of the activities 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 transition End models that the end of the trace has been reached, if no event is to be executed.

Precedence. Direct multiple different activities .

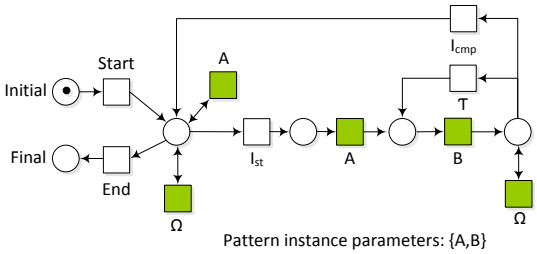

Fig. 30. 'Precedence. Direct or indirect multiple events' Compliance rule

Description: Every activity B must be directly preceded by activity C or A within a chosen scope. If within the specified scope, directly before B one of the activities A or C does not occur, the rule is violated. An instance of this compliance rule includes activity B and its preceding A or C. The Petri-net pattern illustrated in Figure 31 formalizes this rule.

Being in the initial marking, any activity but B may occur. The pattern instance starts as soon as an activity A or C occurs which is followed by B. The structure of the pattern is such that B may occur only if A or C has already occurred before it. After the condition of the rule is satisfied, the *pattern instance* may terminate. The transition End models that the end of the trace has been reached, if no event is to be executed.

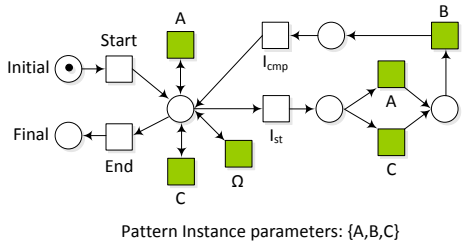

Fig. 31. 'Precedence. Direct multiple different activities' Compliance rule

Precedence. Direct or indirect multiple different activities .
Description: Every activity B must be preceded at least once by activity C or A within a chosen scope. If within the specified scope, before B one of the activities A or C does not occur, the rule is violated. An instance of this compliance rule includes activity B and its preceding A or C. The Petri-net pattern illustrated in Figure 32 formalizes this rule.

The pattern in Figure 32 is similar to the pattern described in Figure 31; with the difference that as long as B is preceded (even indirectly) by any of the activities 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 transition End models that the end of the trace has been reached, if no event is to be executed.

Precedence. Never direct .

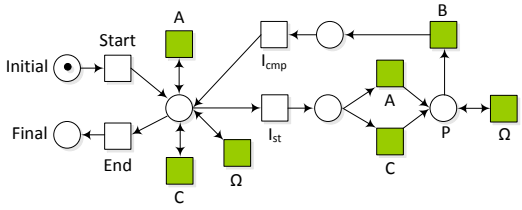

Fig. 32. 'Precedence. Direct or indirect multiple different events' Compliance rule

Description: No activity B must be preceded directly by A within a chosen scope. If A occurs directly before B within the specified scope, the rule is violated. An instance of this compliance rule includes occurrence of all activities A and B and those Ω-labeled activities which occur between occurrences of pair (A, B). The Petri-net pattern illustrated in Figure 33 formalizes this rule.

After the pattern started, any activity may occur. The *pattern instance* triggers as soon as soon as activity A or B occurs. The structure of the pattern should ensure that B cannot occur directly after A. Therefore as soon as first A occurs, place P is marked and B is not enabled anymore. B may occur only after occurrence of an Ω. When the condition of the rule is satisfied, the *pattern instance* completes and the pattern may terminate by firing any of the transitions End, if no event is to be executed.

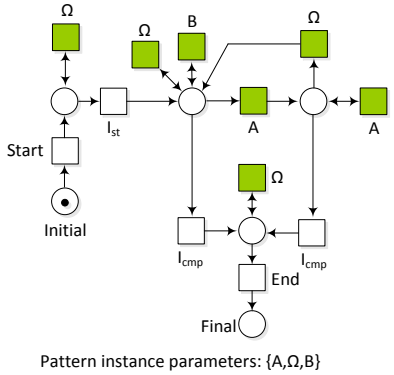

Fig. 33. 'Precedence. Never direct' Compliance rule

Precedence. Never .

Description: Every activity B must never be preceded by A within a chosen scope. If A occurs before B within the specified scope, the rule is violated. An instance of this compliance rule includes occurrence of all activities A and B. The Petri-net pattern illustrated in Figure 34 formalizes this rule.

After the pattern started, any activity may be executed. The *pattern instance* triggers as soon as is triggered as soon as first A or B occurs. After occurrence of the first A, the structure of the pattern should ensure that B cannot occur anymore. Therefore after A occurred, place P is marked and B is not enabled anymore. When the condition of

the rule is satisfied, the *pattern instance* completes and the pattern may terminate firing the transition End, if no event is to be executed.

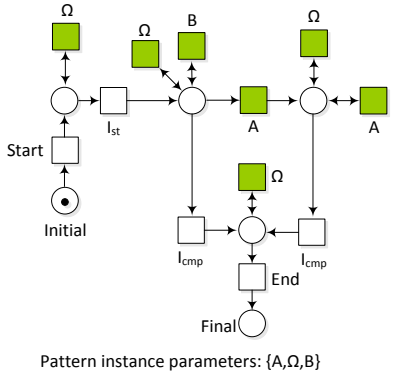

Fig. 34. 'Precedence. Never' Compliance rule

7.7 Chain Precedence Category

This category of compliance rules limits occurrence of a sequence of activities in precedence over another sequence of activities.

Chain Precedence. Direct .
Description: Every sequence of activities $\langle B_1, B_2, \ldots, B_m \rangle$ must be preceded directly by sequence of activities $\langle A_1, A_2, \ldots, A_n \rangle$ within a chosen scope. This rule is violated if within the specified scope, directly before sequence $\langle B_1, B_2, \ldots, B_m \rangle$, sequence $\langle A_1, A_2, \ldots, A_n \rangle$ does not occur. The Petri-net pattern illustrated in Figure 35 formalizes this rule.

This pattern describes the allowed behaviors specified in the rule in *two* cycles. The *pattern instance* is triggered as soon as first A_1 occurs which later leads to occurrence of the sequence $\langle A_1, A_2, \ldots, A_n \rangle$ and is followed directly by the sequence $\langle B_1, B_2, \ldots, B_m \rangle$.

The main cycle of the pattern (please see the shadowed subnet labeled (a)), includes the *pattern instance* (please see the shadowed subnet labeled (b)). When the *pattern instance* is trigged, 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. If the condition of the rule is satisfied, the *pattern instance* completes and the pattern return to the marking where place P is marked. In this marking, other events may be executed occur, another instance of the pattern may start or the pattern may terminate by firing transition End.

However in the main cycle (subnet labeled (a)) if the *pattern instance* is not triggered, from any place 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 marking where place P is marked or terminate the pattern if no activity is to be executed. The return paths are indicated by the smaller cycles inside the main cycle of the pattern.

Being in the marking where P is marked, 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 (c)) 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 marking where there is a token in place P or terminate the pattern if no event is to be executed.

Chain Precedence. Direct or indirect .

Description: Every sequence of activities $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ must be preceded by the sequence of activities $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ within a chosen scope. The rule is violated if within the specified scope, 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 36 formalizes this rule.

The behavior of this pattern is similar to the pattern described in Figure 35, 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).

Chain Precedence. Never direct .

Description: Sequence of activities $\langle B_1, B_2, \ldots, B_m \rangle$ must not be preceded directly by sequence of activities $\langle A_1, A_2, \ldots, A_n \rangle$. The rule is violated if within the specified scope and 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. An instance of this compliance rule includes occurrence of all activities. The Petri-net pattern illustrated in Figure 37 formalizes this rule.

Occurrence of any activity triggers the *pattern instance*. In the main cycle of the pattern (*pattern instance*), 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 marking with a token in the place P 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 marking with a token in P, 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 if $\langle A_1, A_2, \ldots, A_n \rangle$ completes. The *pattern instance* can complete at any point in time if no activity is to be executed and consequently the pattern may terminate.

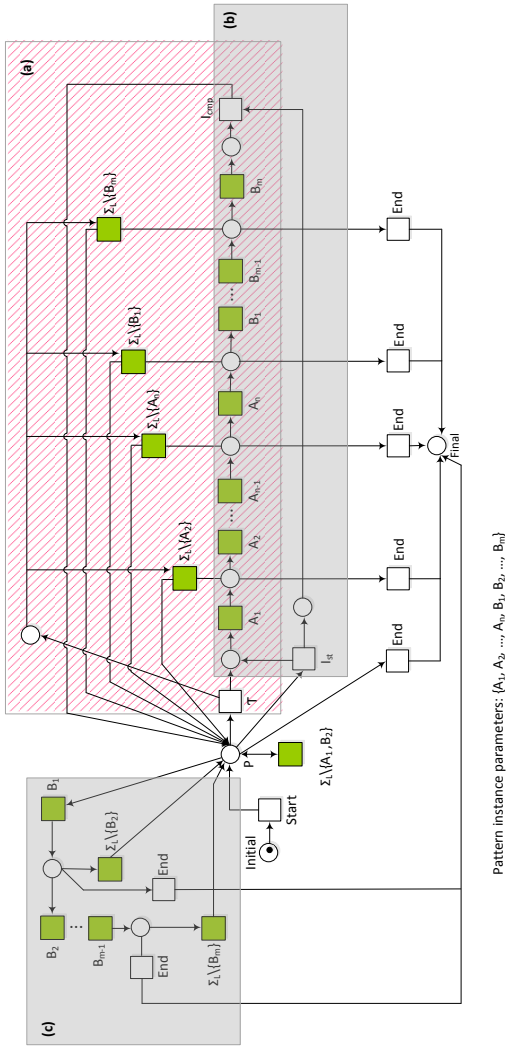

Fig. 35. 'Chain Precedence. Direct' Compliance rule

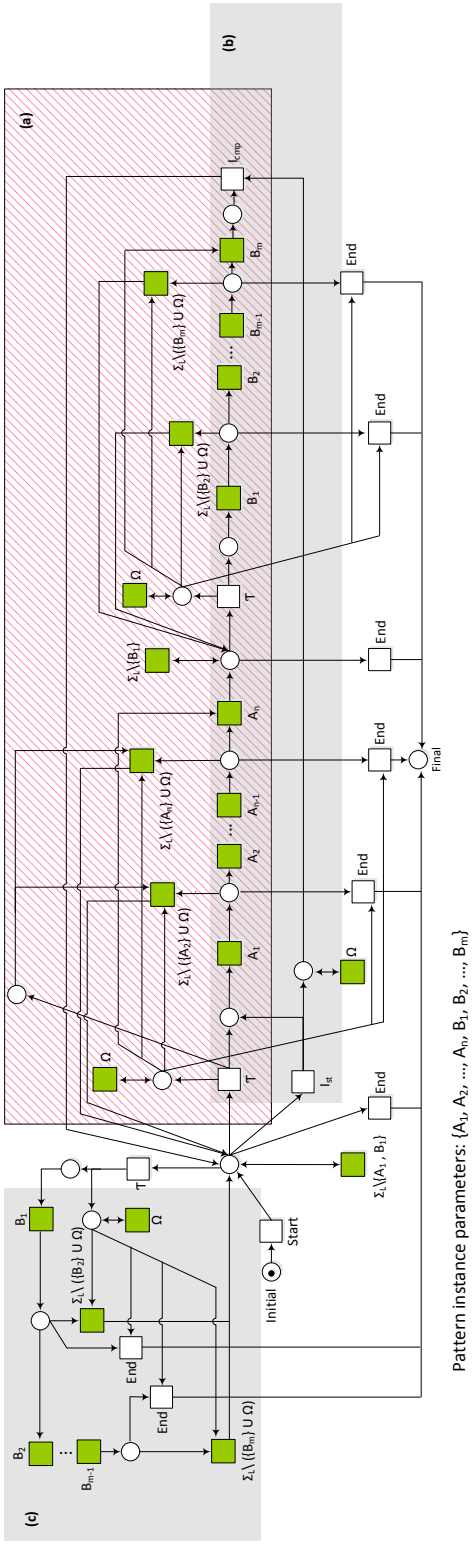

Fig. 36. 'Chain Precedence. Direct or indirect' Compliance rule

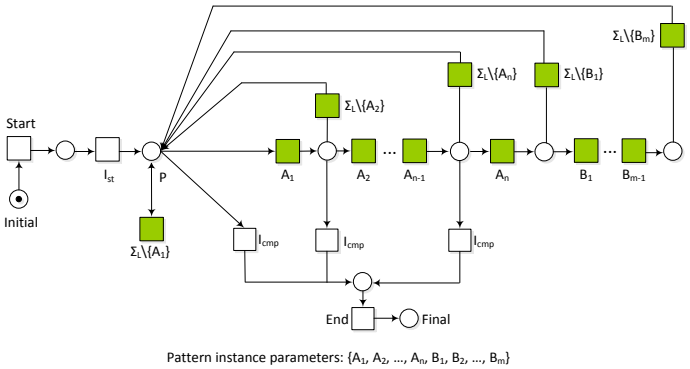

Fig. 37. 'Chain Precedence. Never direct' Compliance rule

Chain Precedence. Never .
Description: Sequence of activities $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ must never be preceded by a sequence of activities $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ within a chosen scope. The rule is violated if within the specified scope and 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. An instance of this compliance rule includes occurrence of all activities. The Petri-net pattern illustrated in Figure 38 formalizes this rule.
The behavior of this pattern is similar to the pattern described in Figure 37, 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 marking where P is marked 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 instance* can terminate at any point in time if no activity is to be executed and consequently the pattern may terminate.

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

7.8 Response Category

This category includes compliance rules that limit the occurrence of one activity in response to another activity.

Response. Simultaneous or after .
Description: Every activity A must be followed directly by activity B or it must occur simultaneously with activity B within a chosen scope. If within the specified scope,

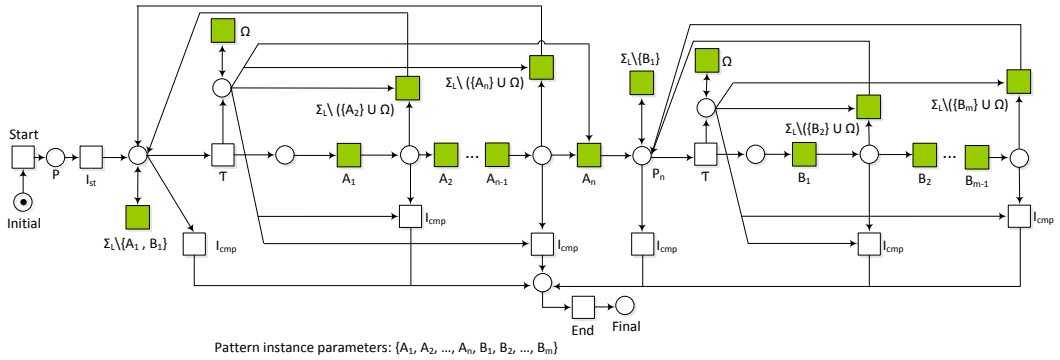

Fig. 38. 'Chain Precedence. Never' Compliance rule

B does not occur directly after A or simultaneously with A, the rule is violated. An instance of this compliance rule includes start and completion of both activities A and B. The Petri-net pattern illustrated in Figure 39 formalizes this rule.

The pattern illustrated in Figure 39 is similar to the pattern described in Figure 25; with the difference in adjacent transitions to the place P. Such that similar to the pattern described in Figure 25, the pattern illustrated in Figure 39 models two options (specified in the rule) for occurrences of A and B. The *pattern instance* starts as soon as A starts. The case where activities 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. When both A and B completes, the condition of the rule is satisfied and the *pattern instance* completes. In this situation, any event may occur, another *pattern instance* may start or the pattern may terminate if no event is to be executed.

In the current compliance rule execution of A, restricts the behavior of the pattern. Therefore B and (Ω)-labeled transitions are adjacent to the place P implying; being in the marking with a token in P, 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 End.

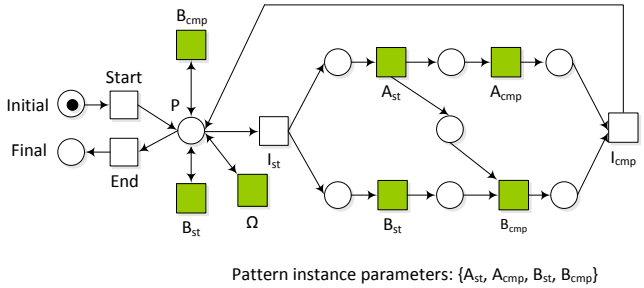

Fig. 39. 'Response. Simultaneous or after' Compliance rule

Response. Direct .

Description: Every activity A must be followed directly by activity B within a chosen scope. If within the specified scope, B does not occur directly after A, the rule is violated. An instance of this compliance rule includes activity A and an activity B which is preceded by A. The Petri-net pattern illustrated in Figure 40 formalizes this rule.

The pattern described in Figure 40 is similar to the pattern described in Figure 26; with the difference in adjacent transitions to the place P. In current pattern, occurrence of event A restricts the behavior of the pattern. Therefore B and (Ω)-labeled transitions are adjacent to the place P implying; being in the marking with a token in place P, if there is no A, B or any (Ω)-labeled transitions may occur.

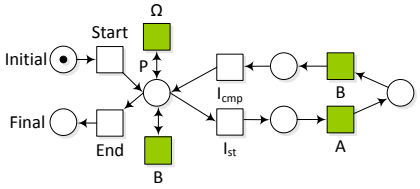

Fig. 40. 'Response. Direct' Compliance rule

Response. Direct or indirect .
Description: Every activity A must be followed eventually by activity B within a chosen scope. If B does not occur after A within the specified scope, the rule is violated. An instance of this compliance rule includes activity A and an activity B which is preceded by A. The Petri-net pattern illustrated in Figure 41 formalizes this rule.

The pattern illustrated in Figure 41 is similar to the pattern described in Figure 40; with the difference that the adjacent Ω-labeled transition to the place P in Figure 41, allows that B indirectly follows any A.

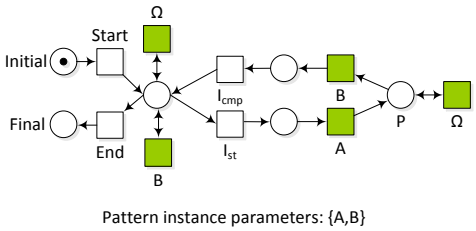

Fig. 41. 'Response. Direct or indirect' Compliance rule

Response. At least once .
Description: Every activity A must always be followed eventually by activity B within a chosen scope. If within the specified scope, B does not occur at least once after A, the rule is violated. An instance of this compliance rule includes all activities A and their following activity B. The Petri-net pattern illustrated in Figure 42 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 42 is similar to the pattern described in Figure

41; with the difference that the adjacent A-labeled transition to place P in Figure 42, allows for arbitrary numbers of occurrences of A. However, eventually B must occur to satisfy the condition of the rule.

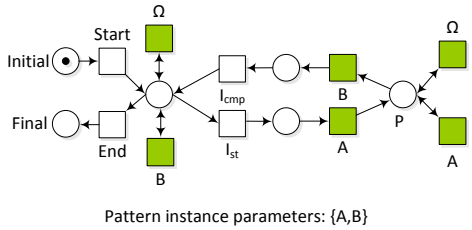

Fig. 42. 'Response. At least once' Compliance rule

Response. Direct multiple activities .
Description: Every activity A must be followed directly by activity B or activity A within a chosen scope. If within the specified scope, directly after A one of the activities B or A does not occur, the rule is violated. An instance of this compliance rule includes an activity B and all activities A preceding it. The Petri-net pattern illustrated in Figure 43 formalizes this rule.

The pattern structure describes that A can only occur if directly after it any of the activities A or B occurs. After the pattern started any event may occur. The *pattern instance* starts as soon as A occurs. A may occur an arbitrary number of times, but the only possibility to complete the *pattern instance* is the occurrence of B. When the condition of the rule is satisfied, the pattern can terminate by firing transition End.

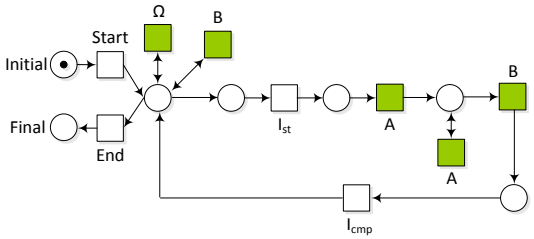

Fig. 43. 'Response. Direct multiple activities' Compliance rule

Response. Indirect multiple activities .
Description: Every activity A must be followed eventually by one of the activities A or B within a chosen scope. If within the specified scope, after A one of the activities B or A does not occur, the rule is violated. An instance of this compliance rule includes an activity B and all occurrences of activity A preceding it. The Petri-net pattern illustrated in Figure 44 formalizes this rule.

The pattern illustrated in Figure 44 is similar to the pattern described in Figure 43; with the difference that as long as A is followed (even indirectly) by any of the activities

A or B, the condition of the rule is satisfied. The adjacent Ω-labeled transition to the place P in Figure 44, allows that A is followed indirectly by A or B.

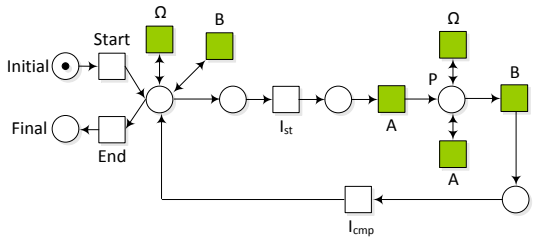

Fig. 44. 'Response. Indirect multiple events' Compliance rule

Response. Direct multiple different activities .
Description: Every activity A must be followed directly by activity B or activity C within a chosen scope. If directly after A one of the activities B or C does not occur, the rule is violated. An instance of this compliance rule includes activity A and its following activity B or C. The Petri-net pattern illustrated in Figure 45 formalizes this rule.

After the pattern started, any activity may occur. The *pattern instance* is triggered as soon as A occurs. 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 activity A without a following B or C. When the condition of the rule is satisfied the *pattern instance* completes and the pattern may terminate. The transition End models that the end of the trace has been reached, if no event is to be executed.

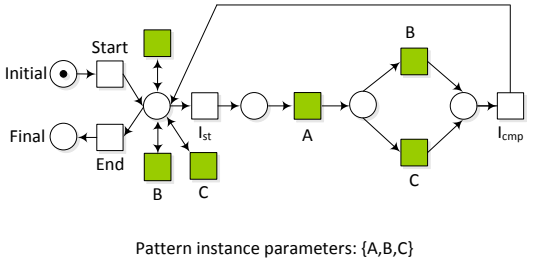

Fig. 45. 'Response. Direct multiple different activities

Response. Indirect multiple different activities .
Description: Every activity A must be followed at least once eventually by activity B or activity C within a chosen scope. If within the specified scope, B or C does not occur at least one time after A, the rule is violated. An instance of this compliance rule includes

activity A and its following activity B or C. The Petri-net pattern illustrated in Figure 46 formalizes this rule.

The pattern illustrated in Figure 46 is similar to the pattern described in Figure 45; with the difference that as long as A is followed (even indirectly) by any of the activities B or C, the condition of the rule is satisfied. The adjacent Ω-labeled transition to the place P in Figure 46, allows that A is followed indirectly by B or C.

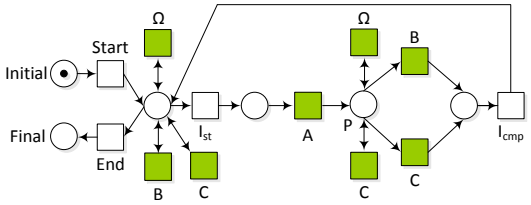

Fig. 46. 'Response. Indirect multiple different activities' Compliance rule

Response. Never direct .

Description: No activity A must be followed directly by Activity B within a chosen scope. If within the specified scope, B occurs directly after A, the rule is violated. An instance of this compliance rule includes occurrence of all activities A and B and the Ω-labeled activities occur between the pair (A, B). The Petri-net pattern illustrated in Figure 47 formalizes this rule.

The *pattern instance* starts with occurrence of any activity. As soon as A occurs, structure of the pattern ensures 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 instance* can complete at any point and consequently the pattern may terminate by firing the transition End.

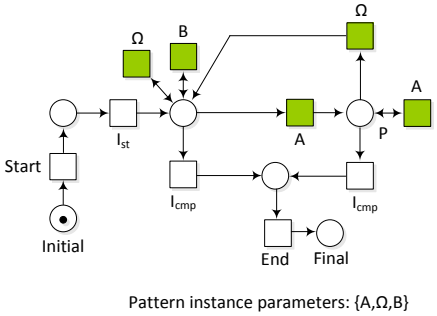

Fig. 47. 'Response. Never direct' Compliance rule

Response. Never .

Description: No activity A must be followed by B within a chosen scope. If within the specified scope, B occurs after A, the rule is violated. An instance of this compliance rule includes occurrence of all activities A and B and the Ω-labeled activities occur between the pair (A, B). The Petri-net pattern illustrated in Figure 48 formalizes this rule.

The *pattern instance* starts with occurrence of any activity. As soon as A occurs, structure of the pattern ensures that B cannot occur after A. Therefore after A occurs, place P is marked and B is not enabled anymore. The *pattern instance* can complete at any point and consequently the pattern may terminate by firing the transition End.

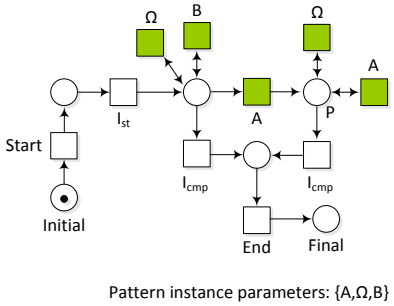

Fig. 48. 'Response. Never' Compliance rule

7.9 Chain Response Category

This category of compliance rules, limits occurrences of a sequence of activities in precedence over another sequence of activities within a chosen scope.

The compliance patterns in this category are similar to the compliance patterns described in the category *Chain Precedence Category* in Section 7.7, with slight differences in termination of the patterns and the transitions enabled after the pattern starts. In the patterns described in Section 7.7, the occurrence of the sequence of activities $\langle B_1, B_2, \ldots, B_m \rangle$ 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 activities $\langle A_1, A_2 \ldots, A_n \rangle$ limits the behavior of the patterns.

Chain Response. Direct .
Description: Every sequence of activities $\langle A_1, A_2, \ldots, A_n \rangle$ must be followed directly by a sequence of activities $\langle B_1, B_2, \ldots, B_m \rangle$ within a chosen scope. The rule is violated if within the specified scope, 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. An instance of this compliance rule includes sequence of $\langle A_1, A_2, \ldots, A_n \rangle$ and its following activities which is specified to be directly the sequence $\langle B_1, B_2, \ldots, B_m \rangle$. The Petri-net pattern illustrated in Figure 49 formalizes this rule.

This pattern describes the allowed behavior specified in the rule in one main cycle. The main cycle of the pattern (subnet labeled (a)) includes the activities of the *pattern instance*(shadowed subnet labeled (b)) as well as occurrences of other activities. The

pattern instance is triggered as soon as first A_1 occurs which later leads to sequence $\langle A_1, A_2, \ldots, A_n \rangle$. The *pattern instance* structure is such 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 (subnet labeled (a)), between place P 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 marking where there is a token in P or terminate the pattern by firing one of the transitions End 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 *pattern instance* starts i.e., as soon as 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. After the completion of the *pattern instance*, any event may occur, another *pattern instance* may start or the pattern may terminate.

In the marking where there is a token in place P, 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.

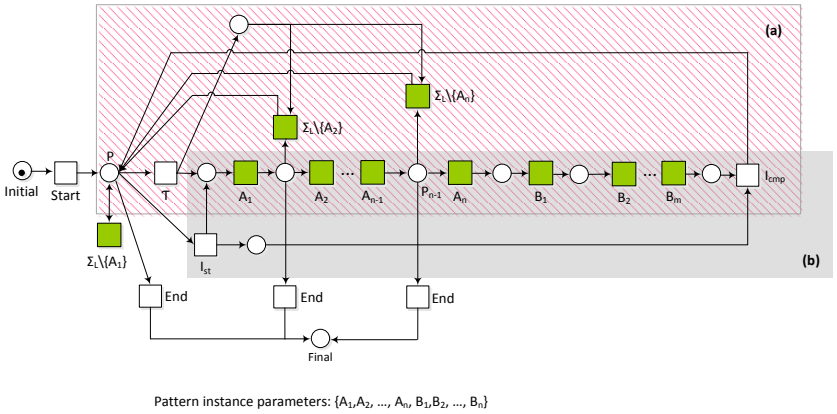

Fig. 49. 'Chain Response. Direct' Compliance rule

Chain Response. Direct or indirect .

Description: Every sequence of activities $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ must be followed eventually by sequence of activities $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ within a chosen scope. The rule is violated if within the specified scope and 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. An instance of this compliance rule includes sequence of $\langle A_1, A_2, \ldots, A_n \rangle$ and its following activities which is specified to be the sequence $\langle B_1, B_2, \ldots, B_m \rangle$. The Petri-net pattern illustrated in Figure 50 formalizes this rule.

The behavior of this pattern is similar to the pattern described in Figure 49, with the difference that both indirect or direct occurrence of sequence $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ after sequence $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ considered to be compliant based on the compliance rule.

This pattern describes the allowed behavior specified in the rule in one main cycle. The main cycle of the pattern (subnet labeled (a)) includes the activities of the *pattern instance* (shadowed subnet labeled (b)) as well as occurrences of other activities. The *pattern instance* is triggered as soon as first A_1 occurs which later leads to sequence $\langle A_1, A_2, \ldots, A_n \rangle$. The *pattern instance* structure is such that the sequence $\langle A_1, A_2, \ldots, A_n \rangle$ can only occur if it is followed eventually by the sequence $\langle B_1, B_2, \ldots, B_m \rangle$.

From any place in the main cycle between place P 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 marking where there is a token in P or terminate the pattern by firing transition End if no event is to be executed. The return paths are indicated with smaller cycles inside the main cycle of the pattern. After the *pattern instance* starts i.e., sequence $\langle A_1, A_2, \ldots, A_n \rangle$ is complete, any activity 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 activity 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 $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$.

After the pattern starts, 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 when $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ completes.

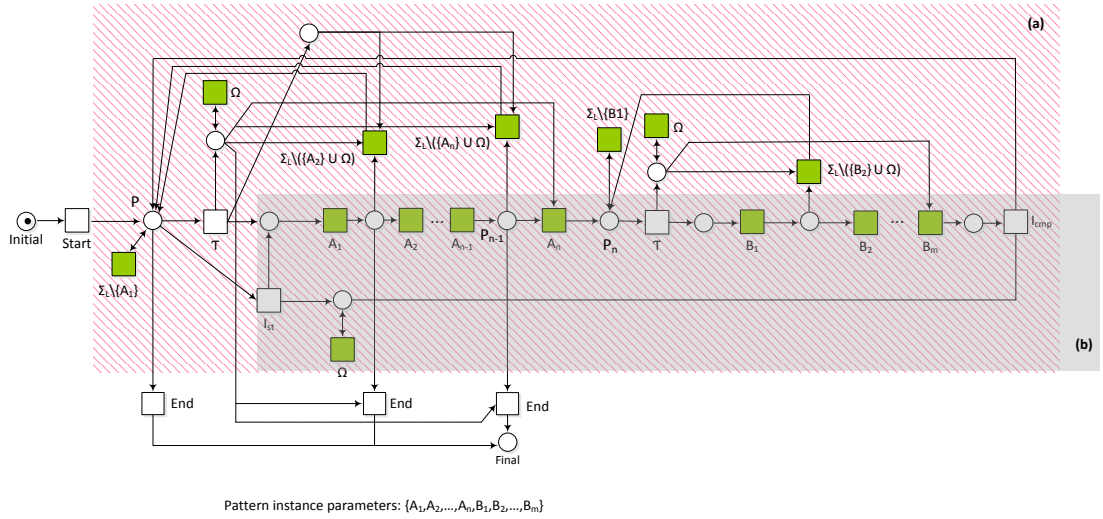

Fig. 50. 'Chain Response. Direct or indirect' Compliance rule

Chain Response. Never direct .

Description: Sequence of activities $\langle A_1, A_2, \ldots, A_n \rangle$ must never be followed directly by sequence of activities $\langle B_1, B_2, \ldots, B_m \rangle$ within a chosen scope . The rule is violated if within the specified scope and 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. An instance of this compliance rule includes occurrences of all activities. The Petri-net pattern illustrated in Figure 51 formalizes this rule.

Occurrence of any activity in the main cycle of the pattern triggers the *pattern instance* start. The *pattern instance* structure is such that 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 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 *pattern instance*, where 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 marking where the place P is marked. The pattern may 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 49.

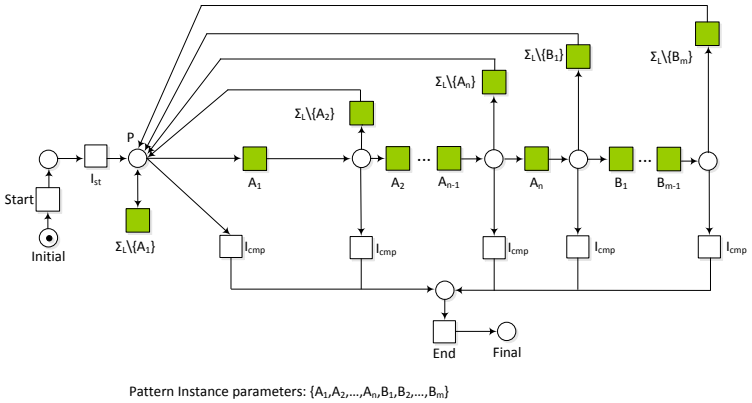

Fig. 51. 'Chain Response. Never direct' Compliance rule

Chain Response. Never .

Description: Sequence of activities $\langle A_1, \ldots, A_2, \ldots, A_n \rangle$ must never be followed by sequence of activities $\langle B_1, \ldots, B_2, \ldots, B_m \rangle$ within a chosen scope. The rule is violated if within the specified scope and 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. An instance of this rule includes occurrences of all activities. The Petri-net pattern illustrated in Figure 38 formalizes the current compliance rule and the *Chain Precedence. Never* compliance rule. The behavior of the pattern is already described in Section 7.7.

7.10 Between Category

This category of compliance rules, limits occurrence of an activity within (between) a sequence od activities within a chosen scope.

Between. After-Before .
Description: Every activity B must always occur after an occurrence of activity A and before an occurrence of activity C within a chosen scope. The rule is violated if within the specified scope, B does not occur between A and C (after A and before C). An instance of this compliance rule includes activity B and its preceding activity A and following activity C. The Petri-net pattern illustrated in Figure 52 formalizes this rule.

After the pattern started any activity but B may occur. As soon as an activity A occurs which is followed by B, the *pattern instance* starts. B can only occur if A has already occurred before it. Moreover B must be followed eventually by C, otherwise there is no possibility that *pattern instance* completes. When the condition of the rule is satisfied, the pattern may terminate by firing transition End.

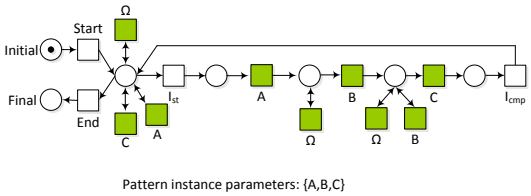

Fig. 52. 'Between. After-Before' Compliance rule

Between. Simultaneously or after-Before .
Description: Every activity B must always occur directly after or simultaneously with an occurrence of activity A and directly before an occurrence of activity C within a chosen scope. The rule is violated if within the specified scope, B does not occur after or simultaneous with A and directly before C. An instance of this compliance rule includes activity B and its preceding or simultaneous activity A and following activity C. The Petri-net pattern illustrated in Figure 53 formalizes this rule.

After the pattern started, any activity may occur. As soon as B starts or an activity A occurs which is followed by B, the *pattern instance* starts. 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 53, which is similar to the structure already described in the pattern in Figure 25.

When activities A and B both complete, the place P is marked. At this marking C must occur, otherwise there is no possibility to complete the *pattern instance*. When the condition of the rule is satisfied, the pattern may terminate by firing transition End.

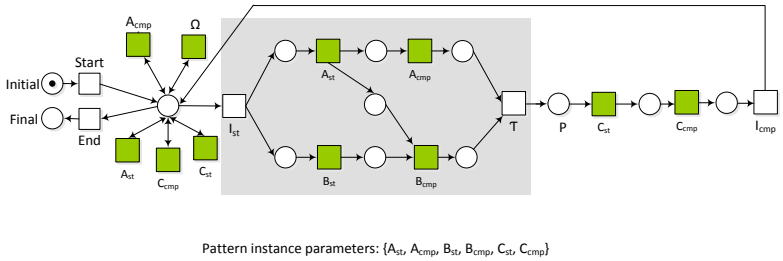

Fig. 53. 'Between. Simultaneously or after-Before' Compliance rule

Between. After-Simultaneously or before .

Description: Every activity B must always occur directly after an occurrence of activity A and directly before or simultaneously with an occurrence of activity C within a chosen scope. The rule is violated if within the specified scope, B does not occur directly after A and directly before or simultaneously with C. An instance of this compliance rule includes activity B and its preceding activity A and following or simultaneous activity C. The Petri net pattern illustrated in Figure 54 formalizes this rule.

After the pattern started any activity but B may occur. As soon as an activity A occurs which is followed by B, the *pattern instance* starts. 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 54, which is similar to the structure already described in the pattern in Figure 25.

The *pattern instance* may complete only if both B and C are completed. When the condition of the rule is satisfied, the pattern may terminate by firing transition End.

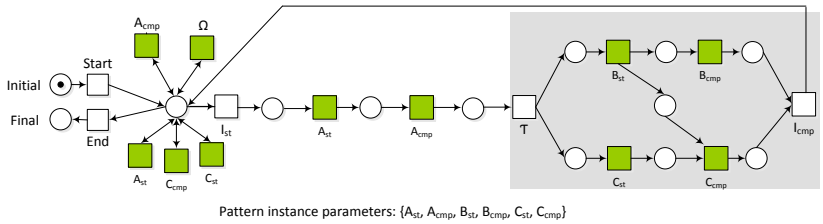

Fig. 54. 'Between. After-Simultaneously or before' Compliance rule

Between. Directly after-Directly before .

Description: Every activity B must always occur directly after activity A and directly before activity C within a chosen scope. The rule is violated if within the specified scope, B does not occur between the sequence of activities $\langle A, C\rangle$. An instance of this compliance rule includes activity B and its directly preceding activity A and directly following activity C. The Petri-net pattern illustrated in Figure 55 formalizes this rule.

This rule specifies the occurrence of the exact sequence of activities $\langle A, B, C\rangle$. After the *pattern instance* starts, occurrence of any activity but B is possible. As soon

as an activity A occurs which is followed directly by B, the *pattern instance* starts. The *pattern instance* may complete only if the exact sequence $\langle A, B, C \rangle$ completes. When the condition of the rule is satisfied, the pattern may terminate by firing transition End.

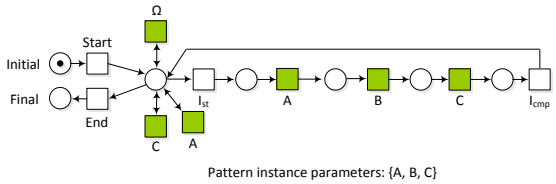

Fig. 55. 'Between. Directly after-Directly before' Compliance rule

Between. Simultaneously-Simultaneously .

Description: Every activity B must always occur simultaneously with activities A and C within a chosen scope. If within the specified scope, activity B does not occur at the same time with occurrence of activities B and C, this compliance rule is violated. An instance of this compliance rule includes activity B and an activity A and an activity C which are executed simultaneously with B. The Petri-net pattern illustrated in Figure 56 formalizes this rule.

After the pattern started, any activity may occur. The *pattern instance* starts as soon as B starts or any of the activities A or C start which are simultaneous with B. After B started, B can only proceed for completion if the activities A and C have already started. When all the activities A, B and C are completed, the *pattern instance* completes. When the condition of the rule is satisfied, the pattern may terminate by firing transition End.

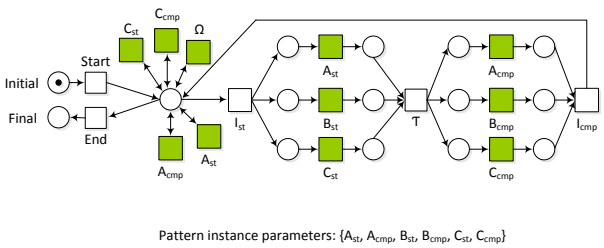

Fig. 56. 'Between. Simultaneously-Simultaneously' Compliance rule

Between. Simultaneously or after-Simultaneously or before .

Description: Every activity B must always occur directly after or simultaneously with activity A and directly before or simultaneously with activity C within a chosen scope. This rule is violated if within the specified scope, B occurs before A or after C or not in the exact sequence of $\langle A, B, C \rangle$. An instance of this compliance rule includes activity B and its directly preceding or simultaneous A and its directly following or simultaneous activity C. The Petri-net pattern illustrated in Figure 57 formalizes this rule.

The pattern described in the Figure 57 is the combination of two simpler patterns described already in the Figure 25 and the Figure 39; with B being the common element in them.

The *pattern instance* starts as soon as B starts or an activity A starts which is followed directly by occurrence of B. 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 activity B with respect to A (simultaneous with A or directly after A) is described in the shadowed subnet labeled (a) in Figure 57, which is similar to the structure described in the pattern in Figure 25.

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 57, which is similar to the structure described in the pattern in Figure 39.

After the patterns started, any activity may occur. This is also the situation where the condition of the rule is satisfied and the pattern may terminate by firing transition End.

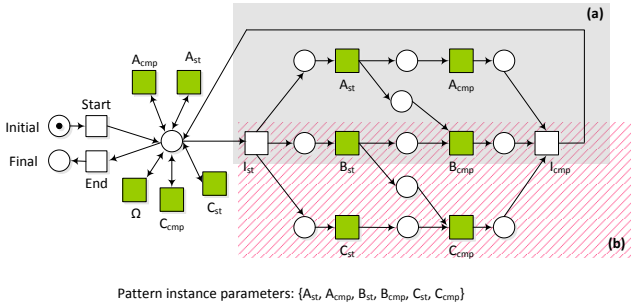

Fig. 57. 'Between. Simultaneously or after-Simultaneously or before' Compliance rule

Between. At least one other activity .

Description: This compliance rule specifies that between every sequence of activities $\langle A, B\rangle$ or $\langle B, A\rangle$ (A before B or B before A) there should be at least one other activity within a chosen scope. This rule is violated if within the specified scope, B occurs directly after A or if A occurs directly after B. An instance of this compliance rule includes activity A, activity B and in case the appear in a sequence, the Ω-labeled activity occur between the sequence $\langle A \ldots B\rangle$. The Petri-net pattern illustrated in Figure 58 formalizes this rule.

After the pattern started any activity may occur. The *pattern instance* is triggered as soon as one of the activities A or B occurs. The right cycle in the pattern ensures that as soon as activity A occurs, it cannot be followed directly by B, i.e., B can only occur if after A at least one other activity (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 activity (Ω) is executed. The pattern can terminate at any point of time if no event is to be executed by firing transition End.

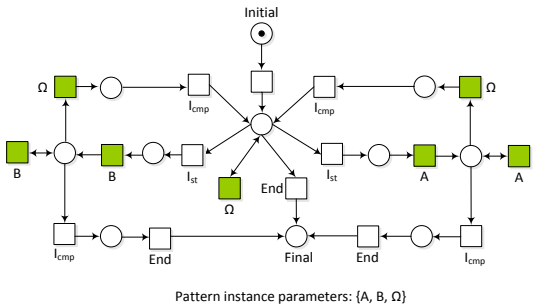

Fig. 58. 'Between. At least one other activity' Compliance rule

8 Separating Temporal Compliance Checking and Control-Flow Compliance Checking

We conducted an extensive literature survey and identified numerous works [28, 38, 27, 38, 50, 30, 6] discussing temporal compliance rules and their formalization. Typically every compliance requirement restricting the process time implies a control flow compliance rule in addition to a temporal compliance rule. Even if the ordering of activities is not restricted in the compliance requirement, at least the existence of some activities is specified. In the general case, the control-flow rule constrains more than just the existence of activities, for instance that "antibiotics must be administered in cycles of 3 occurrences". This leads to a simple assumption for temporal compliance requirements: a temporal compliance rule constrains the occurrences of events specified in a given control-flow rule (e.g., "between two subsequent administration of antibiotics in a cycle, there should be one day delay"); a "larger" temporal rule simply implies a larger control-flow rule.

A control-flow rule may have to hold multiple times in a trace [33], based on repeated occurrences of events. For instance, if antibiotics are administered 6 times in total, the control-flow rule given above has to hold twice; the associated temporal rule has to hold whenever the control-flow rule occurs. The dependency between control-flow rules and temporal rules raises a challenge for temporal compliance checking: we first have to identify the different occurrences of a control-flow rule, for which then the temporal rule can be checked. This gives rise to our approach shown in Fig. 59.

A complex compliance requirement is decomposed into a control-flow rule and a temporal rule. The event log is then first aligned to the control-flow rule using the technique of Sect. 3 to identify control-flow violations in terms of missing or inserted activities; this alignment will also distinguish multiple occurrences of the same control-flow rule within one trace. For each alignment, we then enrich the log with information about multiple occurrences of rules and control-flow violations. Each enriched log is then used to check temporal compliance of the trace using the data-aware alignments of Sect. 3.

However, there is a small challenge in decomposing control-flow and temporal compliance checking. In case a control-flow violation can be attributed to different occurrences of a control-flow rule, there exists more than one alignment of a trace to the control-flow rule. Picking a wrong distribution of control-flow violations to rule occurrences may introduce invalid temporal violations, i.e., false positives. We eliminate

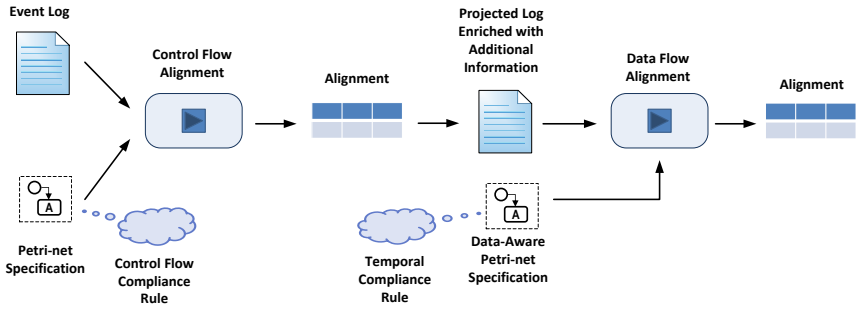

Fig. 59. Temporal Compliance Checking Overview

false positives by computing all best alignments of a trace to the given control-flow rule. The enriched log then contains one *trace variant* for each best alignment of the original trace. After temporal compliance checking, we pick from all variants of an original trace, the variant with the best temporal compliance and eliminate all other variants from the result. The remaining trace variant contains only temporal compliance violations that are real, i.e., cannot be removed by rearranging control-flow violations.

Aligning control-flow and temporal checking has an advantage regarding diagnostic information. A severe control-flow violation implies a violation of the temporal compliance rule. Checking temporal compliance alone might obscure insights into the nature of the violation. By aligning control-flow and temporal compliance checking, we can present more meaningful diagnostic information to a user.

9 Temporal Compliance Checking

Please recall our example from Sect.6. To align temporal compliance checking with control-flow compliance checking, we enrich the original trace σ with information about rule instances and control-flow deviations, as follows. (1) Translate each move of the alignment γ into a log event, where each event originating in a non-synchronous move is marked by a special "move" attribute. (2) Enrich each event of a synchronous move with all attributes of the original event in trace σ. (3) Provide each event of a move on specification with missing attributes, in particular an event without a $time$ attribute gets the $time$ value of the directly preceding event (with the exception of $Start$ and I_{st} which get the $time$ value of the succeeding event).

For example, using alignment γ_1 we enrich trace σ given above to the traces $\sigma_{\gamma,1} = \langle (Start, 1)(B, 1)(I_{st}, 2)(A, 2)(A, 30)(A, 54)(I_{cmp}, 54)(I_{st}, 100)(A, 100)(C, 123)(A, 123, missing)$ $(A, 162)(I_{cmp}, 162)(D, 173)(End, 173)$. This traces now contains enough information to check temporal compliance.

The extended control-flow compliance checking technique produces an enriched log. Each trace of the enriched log distinguishes all different instances of the control-flow rule that underlies the temporal rule to be checked. In addition, log events that violate the control-flow rule are marked in the log. In this section, we present a generic temporal compliance checking technique for logs that are enriched in this way. We first show how to formalize a temporal compliance rule in terms of a *data-aware Petri*

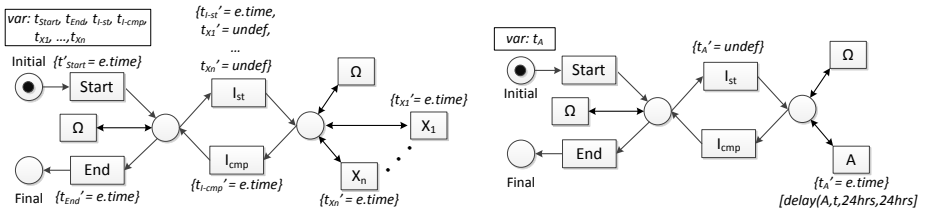

Fig. 60. Temporal Petri-net Pattern, generic (left) and instantiated (right)

net [29] and then how to use data-aware alignments to produce detailed diagnostic information about compliance violations.

9.1 Formalizing Temporal Compliance Rules

We explain the formalization of temporal compliance rules by the first temporal rule of our running example. The concrete temporal rule reads "Between two subsequent administration of antibiotics in a cycle, there should be one day delay," which we abstract to "the delay between two subsequent executions of activity A in an instance of a control-flow rule, must be within $[\alpha, \beta]$ time units." The data-aware Petri net of Fig. 60(right) formalizes this rule.

The Petri net has a very simple control-flow structure that just distinguishes begin and end of a trace (places $Initial$ and $Final$), and whether the trace is *within* an instance of a control-flow rule (after I_{st} occurred) or *outside* a control-flow rule (after I_{cmp} occurred). Transitions labeled Ω allow occurrences of all other activities not constrained by the temporal rule. The actual temporal aspect is described by the variable t_A and the data annotations at transition A and I_{st}. Annotation $\{t'_A = e.time\}$ at A ensures that t_A holds the timestamp of the most recent occurrence of activity A. The most important annotation is the guard $[delay(A, t, \alpha, \beta)]$ defined by $delay(A, \alpha, \beta) \equiv t'_A \in [t_A{+}\alpha, t_A{+}\beta] \vee t_A = undef$. The guard states that the time t'_A of the current occurrence of A has to be in the interval $[t_A + \alpha, t_A + \beta]$, where t_A is the timestamp of the most recent occurrence of A. As the rule only ranges over occurrences of A within the same instance of the control-flow rule, we have to take special care for the first occurrence of A in an instance. The annotation at I_{st} initializes $t_A = undef$ so that the guard of A also holds for the first A. By setting parameters $A =$ *antibiotic administration* and $\alpha = \beta = 24$ hours, the pattern of Fig. 60 formalizes the given temporal rule.

9.2 Checking Temporal Compliance

We check compliance of a trace to the formalized rule on the enriched log trace obtained at the beginning of Sect. 9, for instance trace $\sigma_{\gamma,1}$. The data-aware alignment technique explained in Sect. 3 compares the time stamp of events in $\sigma_{\gamma,1}$ with admissible time stamps defined in the guards of the data-aware Petri net and will give a data-aware alignment γ_1^t, with the least cost, as follows: γ_1^t =

$(Start, 1)$	$(B, 1)$	$(I_{st}, 2)$	$(A, 2)$	...
$(Start, 1)$	$(\Omega, 1)$	$(I_{st}, 2)$	$(A, 2)$	...

...	***(A,30)***	$(A, 54)$	$(I_{cmp}, 54)$	$(I_{st}, 100)$	$(A, 100)$	$(C, 123)$	***(A,123,missing)***	***(A,162)***	$(I_{cmp}, 162)$	$(D, 173)$	$(End, 173)$
...	***(A,26)***	$(A, 54)$	$(I_{cmp}, 54)$	$(I_{st}, 100)$	$(A, 100)$	$(\Omega, 123)$	***(A,124,missing)***	***(A,148)***	$(I_{cmp}, 162)$	$(\Omega, 173)$	$(End, 173)$

.

As is shown in the alignment γ_1^t the second A in the first instance occurred 28 time units after the preceding A, which violates the temporal rule. The data-aware alignment

in addition returns the time at which the event should have occurred at the bottom row of the alignment. In the same way, two deviations in the second iteration are highlighted. However, the "correct" timestamps 124 and 148 suggested by the alignment have to be inspected carefully as in the second instance the second A was missing in the original log (a control-flow violation indicated by the attribute value $missing$). Recall from Sect. 8 that we may have to check several enriched variants of the same original trace (differing in control-flow violations); after checking all variants, the one with the least temporal violations is returned and all other are discarded.

9.3 A Generic Temporal Pattern

We identified 15 generic temporal compliance rules Sect. 10. Each rule can be formalized in a data-aware Petri net similar to Figure 60(right). The generic pattern is shown in Figure 60(left). It permits to constrain occurrences of n generic activities $X_1, \ldots, X_n$, as well as the $Start$ and End of a trace and start and end of each instance (by I_{st} and I_{cmp}). Each formalization of a compliance rule assigns a guard to one or more transitions of the pattern, depending on the particular temporal property. We show some more formalizations next.

The rule "The delay between execution of two subsequent instances of a control-flow rule, must be within $[\alpha, \beta]$ time units." (which expresses the second temporal rule of our running example of Sect. 4.) The formalization of this rule instantiates Figure 60(left) with $n = 0$ (no activity X_i), variables $t_{I_{st}}$ and $t_{I_{cmp}}$ and the guard $delay2(I_{cmp}, I_{st}, \alpha, \beta) \equiv t'_{I_{st}} \in [t_{I_{cmp}} + \alpha, t_{I_{cmp}} + \beta] \vee t_{I_{cmp}} = undef$ assigned to transition $t_{I_{st}}$. This way, I_{st} is only allowed to occur between $t_{I_{cmp}} + \alpha$ and $t_{I_{cmp}} + \beta$ where $t_{I_{cmp}}$ is the last time I_{cmp} occurred (if there was a last occurrence). Checking temporal compliance of $\sigma_{\gamma,1}$ of Sect. 6 to this rule for $\alpha = 7$ days and $\beta = \infty$ (and mapping all activities to Ω), we obtain the following data-aware alignment:
γ_3^t = $\frac{(Start,1)|(B,1)|(I_{st},2)|(A,2)|(A,30)|(A,54)|(I_{cmp},54)|(I_{st},\mathbf{100})|(A,100)|(C,123)|\ldots}{(Start,1)|(\Omega,1)|(I_{st},2)|(\Omega,2)|(\Omega,30)|(\Omega,54)|(I_{cmp},54)|(I_{st},\mathbf{242})|(\Omega,100)|(\Omega,123)|\ldots}$.

The alignment γ_3^t highlights a deviation for the start of the second instance of "3 *administrations of antibiotics*." According to the log, the second administration started just 46 hours after the preceding treatment where the rule requires a delay of at least 1 week (= 168 hours); the correct time is shown in the bottom row of the alignment.

Also compliance rules requiring the absence of an activity in a particular interval can be formalized: "No activity A within all instances of a control flow pattern may be executed within $[\alpha, \beta]$ time units since time t." For this temporal rule, the generic temporal pattern of Fig. 60, has $n = 1$ transition. The guard for this temporal rule is: $negation\_activity\_execution(X_1, t, \alpha, \beta) \equiv t'_{X_1} \notin [t + \alpha, t + \beta]$. Here, the time t can be a fixed time, or the time of some other activity (e.g., include X_2 in the pattern and define $t = t_{X_2}$).

Many temporal compliance requirements found in literature combine several constraints on the relation between the *start* and *completion* of two different activities; for instance, "within all instances of a control flow rule, activity B must start within $[\alpha_{st}, \beta_{st}]$ time units after activity A starts, and activity B must complete within $[\alpha_{cmp}, \beta_{cmp}]$ time units before activity A completes." For this temporal rule, the generic temporal pattern of Figure 60, has $n = 4$ transitions labeled A_{st}, A_{cmp}, B_{st} and B_{cmp} express-

ing the start and completion of A and B, respectively. The pattern uses the generic guard $after(X, Y, \alpha, \beta) \equiv t'_Y \in [t_X + \alpha, t_X + \beta] \vee t_X = undef$ twice: once as $after(A_{st}, B_{st}, \alpha_{st}, \beta_{st})$ at transition B_{st} and once as $after(B_{cmp}, A_{cmp}, \alpha_{cmp}, \beta_{cmp})$ at transition A_{cmp}. Other combinations of this temporal constraint can be expressed in the same way varying the parameters of the guards.

Similarly, all other identified temporal compliance constraints identified in literature can be formalized by instantiating the generic temporal pattern of Figure 60(left); see next section, Sect.10, for details. Each formalization is then eligible for temporal compliance checking using data-aware alignments. Our temporal compliance checking technique is not limited to predefined control-flow rules and temporal rules, but is extendible.

10 Collection of Temporal Compliance Rules and Their Formalization

As explained earlier, the *Generic Temporal Pattern* in Figure 60(left) together with guards formalizes our collection of temporal compliance rules. We collected 15 generic compliance rules from literature [28, 38, 27, 38, 50, 30, 6], and classified them into 7 categories (Please see Table 2).

The *Generic Temporal Pattern* in Figure 60 (left) together with guards specifying temporal rules, follow same principles explained for Control-flow compliance patterns in Sect.7. In addition there are some specific systematics about the temporal Petri-net pattern and temporal compliance rules, we would like to present it at the beginning of this section:

- Occurrences of activity(s) specified in the temporal compliance rule are mimicked respectively by the $X_1, ..., X_n$-labeled transitions. 'n' equals to the number of activity parameters specified in the respective temporal compliance rule.
- Occurrences of any other activities than the activity(s) specified in the temporal compliance rules are mimicked by the Ω-labeled transition. This way, the temporal Petri-net pattern abstract from all other trace activities that are not described in the temporal compliance rule.
- Every transition in the temporal Petri-net pattern (Figure 60(left)) including $\{I_{st}, I_{cmp}, X_1, ..., X_n\}$ has a data attribute assigned to it which stores the time stamp of the events mapped into respective transition. The value of the data attribute (the time stamp of the mapped event) is updated with the occurrence of the next mapped event.
- Occurrence of $\{I_{st}$ writes the value '*Undefined*' over the time stamp of events $\{X_1, ..., X_n\}$.
- Each temporal compliance rule comes with one or several guards. The guard or collection of guards describe all the possible compliant behaviors based on its corresponding temporal compliance rule.
- Guards are specified in terms of a set M. The set M in the temporal Petri-net pattern denotes the set of time stamps which is compliant with the respective temporal compliance rule.

- A trace σ *complies* to a (rule of the) temporal Petri-net pattern if after executing σ, the pattern reaches its *final marking* and all the time stamps of the events in the trace are elements of M.
- Every time interval in temporal compliance rules has a lower bound and an upper bound, indicated by $[\alpha, \beta]$ or $[\gamma, \zeta]$.
- Every time interval in temporal compliance rules starts since time t, where t can be:
  - start of the pattern instance $(t_{I_{st}})$
  - completion of the pattern instance $(t_{I_{cmp}})$
  - start of the process instance (t_{case})
  - start of the calendar $(t_{calendar})$
  - execution time of a specific event (t_{X_i}) within the *pattern instance*.

10.1 Pattern Duration

This category of temporal compliance rules limits the time length in which a control-flow pattern must be executed.

Pattern Duration.Pattern duration *Description*: Every instance of a control flow pattern must be completed within $[\alpha, \beta]$ time units since time t.
Guard:

$pattern\_duration(t, \alpha, \beta) \equiv t'_{I_{cmp}} \in M \ \wedge M = [t + \alpha, t + \beta]$ where t can be chosen by the user from $t \in \{t_{I_{st}}, t_{I_{cmp}}, t_{case}, t_{calendar}, t_{X_{i_{st}}}, t_{X_{i_{cmp}}}\}$
The guard '$pattern\_duration$' is assigned to the I_{cmp} transition in Figure 60(left).

Pattern Duration.Negation pattern duration *Description*: No instance of a control flow pattern must be completed within $[\alpha, \beta]$ time units since time t.
Guard:

$negation\_pattern\_duration(t, \alpha, \beta) \equiv t'_{I_{cmp}} \in M \ \wedge S = [0, \infty) \setminus [t + \alpha, t + \beta]$ where t can be chosen by the user from $t \in \{t_{I_{st}}, t_{I_{cmp}}, t_{case}, t_{calendar}, t_{X_{i_{st}}}, t_{X_{i_{cmp}}}\}$
The guard '$negation\_pattern\_duration$' is assigned to the I_{cmp} transition in Figure 60(left).

10.2 Delay Between Instances

This category of temporal compliance rules limits the delay between two instances of a control-flow rule and it has one temporal compliance rule.
Description: The delay between execution of two instances of a control flow pattern must be within $[\alpha, \beta]$ time units since time t.
Guard: $delay\_between\_instances(t, \alpha, \beta) \equiv t'_{I_{st}} \in M \ \wedge M = [t + \alpha, t + \beta] \ \wedge t \in \{t_{I_{cmp}}\} \ \wedge t_{I_{cmp}} \neq Undefined$

The guard $delay\_between\_instances$ is assigned to the I_{st} transition in Figure 60(left).

10.3 Validity

This category of temporal compliance rules limits the time length in which an activity must be executed or may be executed.

Validity.Activity duration *Description*: Every activity A within all instances of a control flow pattern must be completed within $[\alpha, \beta]$ time units since it starts (time $t_{A_{st}}$).

For this temporal rule, the temporal Petri net pattern in Figure 60(left), has $n = 2$ transitions. Therefore the events A_{st} and A_{cmp} should be respectively mapped to the X_1 and X_2-labeled transitions in the temporal Petri-net pattern.
Guard: $activity\_duration(A_{st}, A_{cmp}, t, \alpha, \beta) \equiv t'_{A_{cmp}} \in M \wedge S = [t+\alpha, t+\beta] \wedge X \neq A_{st} \ \wedge t \in \{t_{A_{st}}\}$
The guard '$activity\_duration$' is assigned to the X_2-labeled transition in Figure 60(left), to which A_{cmp} is mapped.

Validity.Activity execution *Description*: Every activity A within all instances of a control flow pattern must/may be executed within $[\alpha, \beta]$ time units since time t.

For this temporal rule, the temporal Petri net pattern in Figure 60(left), has $n = 1$ transition. Therefore the event A is mapped to the X_1-labeled transitions in the temporal Petri-net pattern.

Please note that this constraint might be associated only to start and completion events of an activity too; in order to restrict only the start or completion of the respective activity (e.g., latest or earliest start date, latest or earliest completion date). If the constraint restricts the start or completion of A, A_{st} or A_{cmp} should be mapped to X_1-labeled transition in the temporal Petri net pattern.
Guard: $activity\_execution(A, t, \alpha, \beta) \equiv t'_A \in M \ \wedge M = [t + \alpha, t + \beta]$ where t can be chosen by the user from $t \in \{t_{I_{st}}, t_{I_{cmp}}, t_{case}, t_{calendar}, t_{X_i}\}$.
The guard '$activity\_execution$' is assigned to the X_1-labeled transition in Figure 60(left) to which the exact event specified in the guard is mapped. This event can be A or A_{st} or A_{cmp} based on the guard. Please note that if $t \in \{t_{X_i}\}$, then the temporal Petri net pattern in Figure 60(left) will have $n = 2$ transitions; Where X_i transition is mapped to the additional transition.

Validity.Negation activity execution *Description*: No activity A within all instances of a control flow pattern may be executed within $[\alpha, \beta]$ time units since time t.

For this temporal rule, the temporal Petri net pattern in Figure 60(left), has $n = 1$ transition. Therefore the event A is mapped to the X_1-labeled transitions in the temporal Petri-net pattern.
Guard: $negation\_activity\_execution(A, t, \alpha, \beta) \equiv t'_A \in M \wedge M = [0, \infty) \setminus [t+\alpha, t+\beta]$ where t can be chosen by the user from $t \in \{t_{I_{st}}, t_{I_{cmp}}, t_{case}, t_{calendar}, t_{X_i}\}$.
The guard '$negation\_activity\_execution$' is assigned to the X_1-labeled transition in Figure 60(left) to which A is mapped. Please note that if $t \in \{t_{X_i}\}$, then the temporal Petri net pattern in Figure 60(left) will have $n = 2$ transitions; Where X_i transition is mapped to the additional transition.

10.4 Time Restricted Existence

This category of temporal compliance rules limits the execution time of a given activity.

Time Restricted Existence.Calendar support existence *Description*: Every activity A within all instances of a control flow pattern may only be executed at times t_1, ..., t_n .

For this temporal rule, the temporal Petri net pattern in Figure 60(left) has $n = 1$ transition. Therefore the event A is mapped to the X_1-labeled transition in the temporal Petri-net pattern.

Please note that this constraint might be associated only to start and completion events of an activity too; in order to restrict only the start or completion of the respective activity (e.g., an activity must start only at times t_1, ..., t_n).

If the constraint restricts the start or completion of A, A_{st} or A_{cmp} should be mapped to X_1-labeled transition in the temporal Petri net pattern.
Guard: $calendar\_support\_existence(A, t_1, .., t_n, \alpha, \beta) \equiv t'_A \in M \wedge M = \{t_1, ..., t_n\}$
The guard '$calendar\_support\_existence$' is assigned to the X_1-labeled transition in Figure 60(left) to which the exact event specified in the guard is mapped. This event can be A or A_{st} or A_{cmp} based on the guard.

Time Restricted Existence.Calendar support absence *Description*: Every A within all instances of a control flow pattern must not be executed at times t_1, ..., t_n .

For this temporal rule, the temporal Petri net pattern in Figure 60(left), has $n = 1$ transition. Therefore the event A is mapped to the X_1-labeled transitions in the temporal Petri-net pattern.

Please note that this constraint might be associated only to start and completion events of an activity too; in order to restrict only the start or completion of the respective activity (e.g., an activity must not start at times t_1, ..., t_n).

If the constraint restricts the start or completion of A, A_{st} or A_{cmp} should be mapped to X_1-labeled transition in the temporal Petri net pattern.
Guard: $calendar\_support\_absense(A, t_1, .., t_n, \alpha, \beta) \equiv t'_A \in M \wedge M = [0, \infty) \setminus \{t_1, ..., t_n\}$
The guard '$calendar\_support\_absense$' is assigned to the X_1-labeled transition in Figure 60(left), to which the exact event specified in the guard is mapped. This event can be A or A_{st} or A_{cmp} based on the guard.

10.5 Repetition Time Constraint

This category of temporal compliance rules limits the repetition of instances of a control flow pattern or an activity over time.

Repetition Time Constraint.Delay Between Activities of one kind *Description*: The delay between execution of two subsequent activities A in all instances of a control flow pattern, must be within $[\alpha, \beta]$ time units since time t.

For this temporal rule, the temporal Petri net pattern in Figure 60(left) has $n = 1$ transition. Therefore the event A is mapped to the X_1-labeled transitions in the temporal Petri-net pattern.
Guard: $delay\_between\_activities(A, t, \alpha, \beta) \equiv t'_A \in M \ \wedge M = [t + \alpha, t + \beta] \ \wedge t \in \{t_A\} \ \wedge t_A \neq Undefined$
The guard '$delay\_between\_activities$' is assigned to the X_1-labeled transition in Figure 60(left), to which A is mapped.

Repetition Time Constraint.Delay Between Activities of different kinds *Description*: The delay between execution of two subsequent activities A and B in all instances of a control flow pattern, must be within $[\alpha, \beta]$ time units since time t.

For this temporal rule, the temporal Petri net pattern in Figure 60(left) has $n = 2$ transitions. Therefore the events A and B are respectively mapped to the X_1-labeled and X_2-labeled transitions in the temporal Petri-net pattern.
Guard: $delay\_between\_different\_activities_1(A, B, t, \alpha, \beta) \equiv t'_A \in M \ \wedge M = [t + \alpha, t + \beta] \ \wedge t \in \{t_B\} \ \wedge t_B \neq Undefined$
The guard '$delay\_between\_different\_activities_1$' is assigned to the X_1-labeled transition in Figure 60(left) to which A is mapped.
Guard: $delay\_between\_different\_activities_2(A, B, t, \alpha, \beta) \equiv t'_B \in M \ \wedge M = [t + \alpha, t + \beta] \ \wedge t \in \{t_A\} \ \wedge t_A \neq Undefined$
The guard '$delay\_between\_different\_activities_2$' is assigned to the X_2-labeled transition in Figure 60(left) to which B is mapped.

Please note that we can limit the *start* of one activity to *completion* of another activity, in this case we can use the same guard explained above with this difference that the parameters of the guard would be instantiated for *start* event of one activity and *completion* event of another activity.

10.6 Time Dependent Variability

This category of temporal compliance rules includes one temporal compliance rule and it specifies that the control flow of the process will vary during execution with respect to time aspects.
Description: Within all instances of a control flow pattern, activity B must be executed within $[\alpha, \beta]$ time units since time t' if A has occurred within $[\gamma, \zeta]$ time units since time t''.

For this temporal rule, the temporal Petri net pattern in Figure 60(left) has $n = 2$ transitions. Therefore the events A and B are respectively mapped to the X_1-labeled and X_2-labeled transitions in the temporal Petri-net pattern.

Guard:

$time\_dependent\_variability(A, B, t', t'', \alpha, \beta, \gamma, \zeta) \equiv t'_B \in M \wedge M = [t' + \alpha, t' + \beta] \ \wedge t'_A \in [t'' + \gamma, t'' + \zeta]$
where t' and t'' can be chosen by the user from $t' \in \{t_{I_{st}}, t_{I_{cmp}}, t_{Start}, t_{Calendar}, t_{X_i}\}$ and $t'' \in \{t_{I_{st}}, t_{I_{cmp}}, t_{Start}, t_{Calendar}, t_{X_i}\}$
The guard $time\_dependent\_variability$ is assigned to the X_2-labeled transition in Figure 60(left) to which B is mapped. Please note that if $t' \in \{t_{X_i}\}$, or $t'' \in \{t_{X_i}\}$ then the

temporal Petri net pattern in Figure 60(left) will have more than 1 transitions; Where X_i transition is mapped to the additional transition.

10.7 Overlap

This category of temporal compliance rules, limits the start and completion of one activity (A) to start and completion of another activity (B). The rules in this category are an extension of the '*Repetition Time Constraint.Delay Between Activities of different kinds*' temporal rule which is explained in Section 10.5. For this temporal category, the temporal Petri net pattern in Figure 60(left), has $(n = 4)$ transitions. Therefore the events A_{st}, A_{cmp}, B_{st} and B_{cmp} are mapped respectively into X_1, ..., X_4-labeled transitions in the temporal Petri-net pattern.

Overlap.Start after, complete before *Description*: Within all instances of a control flow pattern, activity B must start within $[\alpha, \beta]$ time units after activity A starts and activity B must complete within $[\gamma, \zeta]$ time units before activity A completes. This temporal rule has two guards:

$start\_after\_complete\_before_1(A_{st}, B_{st}, t, \alpha, \beta) \equiv\equiv t'_{B_{st}} \in M \wedge M = [t+\alpha, t+\beta] \wedge t \in \{t_{A_{st}}\} \wedge \{t_{A_{st}}\} \neq Undefined.$

The guard $start\_after\_complete\_before_1$ is assigned to X_3 transition in Figure 60(left) into which B_{st} is mapped.

$start\_after\_complete\_before_2(A_{cmp}, B_{cmp}, t, \gamma, \zeta) \equiv t'_{A_{cmp}} \in M \wedge M = [t+\gamma, t+\zeta] \wedge t \in \{t_{B_{cmp}}\} \wedge \{t_{B_{cmp}}\} \neq Undefined$

The guard $start\_after\_complete\_before_2$ is assigned to the X_2-labeled transition in Figure 60(left) to which A_{cmp} is mapped.

Overlap.Start before, complete before *Description*: Within all instances of a control flow pattern, activity B must start within $[\alpha, \beta]$ time units before activity A starts and activity B must complete within $[\gamma, \zeta]$ time units before activity A completes. This temporal rule has two guards:

$start\_before\_complete\_before_1(A_{st}, B_{st}, t, \alpha, \beta) \equiv t'_{A_{st}} \in M \wedge M = [t+\alpha, t+\beta] \wedge t \in \{t_{B_{st}}\} \wedge \{t_{B_{st}}\} \neq Undefined$

The guard $start\_before\_complete\_before_1$ is assigned to the X_1-labeled transition in Figure 60(left) into which A_{st} is mapped.

$start\_after\_complete\_before_2(A_{cmp}, B_{cmp}, t, \gamma, \zeta) \equiv t'_{A_{cmp}} \in M \wedge M = [t+\gamma, t+\zeta] \wedge t \in \{t_{B_{cmp}}\} \wedge \{t_{B_{cmp}}\} \neq Undefined$

The guard $start\_after\_complete\_before_2$ is assigned to the X_2-labeled transition in Figure 60(left) into which A_{cmp} is mapped.

Overlap.Start after, complete after *Description*: Within all instances of a control flow pattern, activity B must start within $[\alpha, \beta]$ time units after activity A starts and activity B must complete within $[\gamma, \zeta]$ time units after activity A completes. This temporal rule has two guards:

$start\_after\_complete\_after_1(A_{st}, B_{st}, t, \alpha, \beta) \equiv t'_{B_{st}} \in M \ \wedge M = [t + \alpha, t + \beta] \ \wedge t \in \{t_{A_{st}}\} \ \wedge \{t_{A_{st}}\} \neq \mathit{Undefined}$

The guard $start\_after\_complete\_after_1$ is assigned to the X_3-labeled transition in Figure 60(left) into which B_{st} is mapped.

$start\_after\_complete\_after_2(A_{cmp}, B_{cmp}, t, \gamma, \zeta) \equiv t'_{B_{cmp}} \in M \ \wedge M = [t + \gamma, t + \zeta] \ \wedge t \in \{t_{A_{cmp}}\} \ \wedge \{t_{A_{cmp}}\} \neq \mathit{Undefined}$

The guard $start\_after\_complete\_after_2$ is assigned to the X_4-labeled transition in Figure 60(left) into which B_{cmp} is mapped.

Overlap.Start before, complete after *Description*: Within all instances of a control flow pattern, activity B must start within $[\alpha, \beta]$ time units before activity A starts and activity B must complete within $[\gamma, \zeta]$ time units after activity A completes. This compliance rule is indeed similar to the compliance rule explained in Section 10.7 with the difference that position of the parameters of the rule is swapped. This temporal rule also has two guards:

$start\_before\_complete\_after_1(A_{st}, B_{st}, t, \alpha, \beta) \equiv t'_{A_{st}} \in M \ \wedge M = [t + \alpha, t + \beta] \ \wedge t \in \{t_{B_st}\} \ \wedge \{t_{B_{st}}\} \neq \mathit{Undefined}$

The guard $start\_before\_complete\_after_1$ is assigned to the X_1-labeled transition in Figure 60(left) into which A_{st} is mapped.

$start\_before\_complete\_after_2(A_{cmp}, B_{cmp}, t, \gamma, \zeta) \equiv t'_{B_{cmp}} \in M \ \wedge M = [t + \gamma, t + \zeta] \ \wedge t \in \{t_{A_{cmp}}\} \ \wedge \{t_{A_{cmp}}\} \neq \mathit{Undefined}$

The guard $start\_before\_complete\_after_2$ should be assigned to the X_4-labeled transition in Figure 60(left) into which B_{cmp} is mapped.

11 Implementation in ProM

Our temporal compliance checking technique is implemented in Process Mining Toolkit ProM, available from `www.promtools.org`, and is validated on several real-life logs.

The package *Compliance* provides 2 user-friendly plugins for control-flow and temporal compliance checking. The first plugin provides control-flow compliance checking as described in Sect. 6: it takes as input a log and returns compliance diagnostics in form of an alignment, the control-flow rule to check compliance for is picked by the user from a rule repository (explained in Sect. 7) using a wizard. The second plugin takes the control-flow alignment, produces an enriched log and then checks temporal compliance of the log to a temporal rule that can be specified by the user through a wizard. The resulting alignment then provides diagnostic information by showing control-flow compliance violations and temporal violations projected into the events of the original log.

12 Conclusion

In this report we presented a generic approach for backwards compliance checking of control-flow and temporal compliance rules which enables us to provide diagnostic information in case of violations.

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. Therefore we presented a comprehensive collection of control-flow compliance and temporal compliance rules. We managed to keep the control-flow perspective and temporal perspective orthogonal. By combining both perspectives we can answer a broad range of compliance questions.

We provide a repository of patterns and the *Compliance* package of ProM implements the approach presented in this work. The software has been tested on various real-life logs. Future research aims at making the approach more user friendly. Now it is not easy to choose from the different compliance rules and instantiate the patterns. Hence, higher-level compliance languages and more intuitive diagnostics are needed for end users.

In addition we would like to explore the compliance rule framework (Fig. 2) further and extending the compliance rule repositories for other dimensions, i.e., with collections of compliance rules restricting data flow, process resource.

Acknowledgements. We thank M. de Leoni, 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. Abid, N., Dal-Zilio, S., Botlan, D.L.: Real-time specification patterns and tools. In: Stoelinga, M., Pinger, R. (eds.) FMICS. Lecture Notes in Computer Science, vol. 7437, pp. 1–15. Springer (2012)
7. 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)
8. Adriansyah, A., Sidorova, N., Dongen, B.F.v.: Cost-based Fitness in Conformance Checking. In: ACSD 2011. pp. 57–66. IEEE (2011)
9. 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)
10. 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)
11. Awad, A., Weske, M.: Visualization of compliance violation in business process models. In: Business Process Management Workshops. pp. 182–193 (2009)

12. Calders, T., Guenther, C., Pechenizkiy, M., Rozinat, A.: Using Minimum Description Length for Process Mining. In: SAC 2009. pp. 1451–1455. ACM Press (2009)
13. 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)
14. 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)
15. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: ICSE 1999. pp. 411–420 (1999)
16. Eder, J., Panagos, E., Rabinovich, M.: Time constraints in workflow systems. In: Jarke, M., Oberweis, A. (eds.) CAiSE. Lecture Notes in Computer Science, vol. 1626, pp. 286–300. Springer (1999)
17. 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)
18. Fötsch, D., Pulvermüller, E., Rossak, W.: Modeling and verifying workflow-based regulations. In: ReMo2V 2006. CEUR Workshop Proceedings, vol. 241 (2007)
19. Ge, N., Pantel, M., Crégut, X.: Formal specification and verification of task time constraints for real-time systems. In: Margaria, T., Steffen, B. (eds.) ISoLA (2). Lecture Notes in Computer Science, vol. 7610, pp. 143–157. Springer (2012)
20. Ghose, A., Koliadis, G.: Auditing Business Process Compliance. In: ICSOC. Lecture Notes in Computer Science, vol. 4749, pp. 169–180 (2007)
21. 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)
22. Goedertier, S., Martens, D., Vanthienen, J., Baesens, B.: Robust Process Discovery with Artificial Negative Events. Journal of Machine Learning Research 10, 1305–1340 (2009)
23. Gruhn, V., Laue, R.: Patterns for timed property specifications. Electr. Notes Theor. Comput. Sci. 153(2), 117–133 (2006)
24. 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)
25. Kharbili, M.: Business process regulatory compliance management solution frameworks: A comparative evaluation. In: APCCM 2012. CRPIT, vol. 130, pp. 23–32. ACS (2012)
26. Koymans, R.: Specifying real-time properties with metric temporal logic. Real-Time Systems 2(4), 255–299 (1990)
27. Lanz, A., Weber, B., Reichert, M.: Time patterns in process-aware information system - a pattern based analysis - revised version. Tech. Rep. UIB-2009, University of Ulm, Germany (2009)
28. Lanz, A., Weber, B., Reichert, M.: Workflow time patterns for process-aware information systems. In: Bider, I., Halpin, T.A., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) BMMDS/EMMSAD. Lecture Notes in Business Information Processing, vol. 50, pp. 94–107. Springer (2010)
29. Leoni, M., Aalst, W., Dongen, B.: Data- and resource-aware conformance checking of business processes. In: Abramowicz, W., Kriksciuniene, D., Sakalauskas, V. (eds.) Business Information Systems, Lecture Notes in Business Information Processing, vol. 117, pp. 48–59. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-30359-3_5

30. Li, H., Yang, Y.: Verification of temporal constraints for concurrent workflows. In: Yu, J.X., Lin, X., Lu, H., Zhang, Y. (eds.) APWeb. Lecture Notes in Computer Science, vol. 3007, pp. 804–813. Springer (2004)
31. Li, H., Yang, Y.: Dynamic checking of temporal constraints for concurrent workflows. Electronic Commerce Research and Applications 4(2), 124–142 (2005)
32. Lu, R., Sadiq, S.W., Governatori, G.: Compliance aware business process design. In: BBM Workshops 2007. LNCS, vol. 4928, pp. 120–131. Springer (2008)
33. Ly, L.T., Rinderle-Ma, S., Knuplesch, D., Dadam, P.: Monitoring business process compliance using compliance rule graphs. In: OTM'11. LNCS, vol. 7044, pp. 82–99. Springer (2011)
34. 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)
35. Montali, M.: Specification and Verification of Declarative Open Interaction Models - A Logic-Based Approach, Lecture Notes in Business Information Processing, vol. 56. Springer (2010)
36. Munoz-Gama, J., Carmona, J.: A Fresh Look at Precision in Process Conformance. In: BPM 2010. LNCS, vol. 6336, pp. 211–226. Springer (2010)
37. Munoz-Gama, J., Carmona, J.: Enhancing Precision in Process Conformance: Stability, Confidence and Severity. In: CIDM 2011. IEEE (2011)
38. Niculae, C.C.: Time patterns in workflow management systems. Tech. Rep. BPM-11-04,, BPM Center Report, BPMcenter.org (2011)
39. Pitzmann, B., Powers, C., Waidner, M.: Ibm's unified governance framework (ugf). Tech. rep., IBM Research Division, Zurich (2007)
40. Pozewaunig, H., Eder, J., Liebhart, W.: epert: Extending pert for workflow management system. In: ADBIS. pp. 217–224. Nevsky Dialect (1997)
41. Ramezani, E., Fahland, D., Aalst, W.: Where did i misbehave? diagnostic information in compliance checking. In: Barros, A., Gal, A., Kindler, E. (eds.) Business Process Management, Lecture Notes in Computer Science, vol. 7481, pp. 262–278. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-32885-5_21
42. 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)
43. Rozinat, A., Aalst, W.M.P.v.d.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. 33(1), 64–95 (2008)
44. 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)
45. 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)
46. 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)
47. 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)
48. 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)
49. 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)

50. Westergaard, M., Maggi, F.M.: Looking into the future: Using timed automata to provide a priori advice about timed declarative process models. In: Proceedings of the 20th International Conference on Cooperative Information Systems (CoopIS 2012). Berlin: Springer (September 2012 2012)
51. Wolter, C., Meinel, C.: An approach to capture authorisation requirements in business processes. Requir. Eng. 15(4), 359–373 (2010)