Checking Regulatory Compliance: Will We Live to See It?

Conference Paper · September 2019

CITATION
1

READS
192

3 authors:

Silvano Colombo Tosatto
20 PUBLICATIONS **59** CITATIONS
SEE PROFILE

Guido Governatori
The Commonwealth Scientific and Industrial Research Organisation
390 PUBLICATIONS **7,081** CITATIONS
SEE PROFILE

Nick R. T. P. van Beest
Data61 | CSIRO, Brisbane
33 PUBLICATIONS **238** CITATIONS
SEE PROFILE

Checking Regulatory Compliance: Will We Live to See It?

Silvano Colombo Tosatto[1], Guido Governatori[1], and Nick van Beest[1]

1: DATA61, CSIRO, Australia,
silvano.colombo, guido.governatori, nick.vanbeest@data61.csiro.au

Abstract. Checking regulatory compliance of business processes prior to deployment is common practice and numerous approaches have been developed over the last decade. However, the computational complexity of the problem itself has never received any major attention. Although it is known that the complexity of the problem is generally in **NP**-complete, many existing approaches ignore the issue using the excuse that current problems are small enough to be solved anyway. However, due to the current race towards digitalisation and automatisation, the size and complexity of the problems is bound to increase. As such, this paper investigates the computational complexity of all different sub-classes of the problem and categorises some of the existing approaches, providing a detailed overview of the issues that require to be tackled in order for current compliance checking solutions to remain feasible in future scenarios.

Keywords: Business Process Modelling, Regulatory Compliance, Computational Complexity

1 Introduction

Recently more and more effort is put into digitalising and automatising businesses and services in various sectors. Considering both the importance and complexity of proving regulatory compliance of such automated business models, studying and understanding the details about the computational complexity of such problems becomes of paramount importance.

Current approaches aiming at proving regulatory compliance of business process models (i.e. verifying whether their executions follow the required regulations) already exist. However, due to the limited size and complexity of the models currently used, the computational complexity of these problems has not yet been an insurmountable issue. Nevertheless, due to the increase in automation and digitalisation, the size of the problems and their complexity are also bound to increase. This, in turn, will lead the computational complexity to be an issue in the future.

With this in mind, we study the computational complexity of some variants of the problem identified by considering some properties, namely the number of regulations to be verified, their expressivity and whether they affect the process partially or entirely. Given these properties, we identify 8 variants. The computational complexity for some of these variants has already been studied by Colombo Tosatto et al. [22,23], showing that the upper bound complexity of the variants of the problem considered in the present paper is **NP**-complete, while the lower bound is in **P**.

While being aware that the regulatory framework is not the only source of computational complexity in the regulatory compliance problem, we have decided to explore and study the variants of the problem obtainable by considering a relatively simple business process model, acyclic structured business processes, and varying the properties of the regulatory framework. Although the computational complexity study provided in this paper does not investigate the whole problem, it still provides an analysis of many variants of the problem. As such, it can be used in the future to start a further computational complexity analysis, such as for instance how the properties of a business process model contribute to the computational complexity of the problem.

Additionally, we analyse some of the existing solutions, and assign them to the variants of the problem identified in this paper. Our goal is to show in this way, which of the currently available solutions are bound to hit the computational complexity wall as automatising compliance proving gains popularity.

The remainder of this paper is structured as follows. Section 2 introduces the problem of proving regulatory compliance of business process model and its semantics. Subsequently, Section 3 introduces the classification and acronyms of the variants of the problem studied in this paper, and discusses some of the existing computational complexity results. Next, Section 4 extends the computational complexity results and provides additional ones for some of the variants. Section 5 shows how some of the existing approaches aiming at solving the regulatory compliance problem can be assigned to the problem's variants identified and studied. Finally, Section 6 concludes the paper. In order to preserve the readability of the paper, we have collected the computational complexity proofs in the Appendix.

2 Proving Regulatory Compliance

In this section, we introduce the problem of proving regulatory compliance of business process models analysed in this paper. The problem consists of two components: i) the business process model compactly describing a set of possible executions, and ii) a regulatory framework, describing the compliance requirements.

2.1 Structured Business Processes

We limit our study to structured process models, as the soundness[1] of such models can be verified in polynomial time with respect to their size, and allows us to reuse some existing results to prove our claims. These processes are similar to structured workflows defined by Kiepuszewski et al. [15].

Definition 1 (Process Block). *A process block B is a directed graph: the nodes are called* elements *and the directed edges are called* arcs*. The set of elements of a process block are identified by the function $V(B)$ and the set of arcs by the function $E(B)$. The set of elements is composed of tasks and coordinators. There are 4 types of coordinators:* and_split*,* and_join*,* xor_split *and* xor_join*. Each process block B has two distinguished nodes called the* initial *and* final *element. The initial element has no incoming arc from other elements in B and is denoted by $b(B)$. Similarly the final element has no outgoing arcs to other elements in B and is denoted by $f(B)$.*

A directed graph composing a process block is defined inductively as follows:

[1] A process is sound, as defined by van der Aalst [1,24], if it avoids livelocks and deadlocks.

– *A single task constitutes a process block. The task is both initial and final element of the block.*
– *Let $B_1, \ldots, B_n$ be distinct process blocks with $n > 1$:*
  - $\mathsf{SEQ}(B_1, \ldots, B_n)$ *denotes the process block with node set* $\bigcup_{i=0}^{n} V(B_i)$ *and edge set* $\bigcup_{i=0}^{n}(E(B_i) \cup \{(f(B_i), b(B_{i+1})) : 1 \leq i < n\})$.
  The initial element of $\mathsf{SEQ}(B_1, \ldots, B_n)$ *is* $b(B_1)$ *and its final element is* $f(B_n)$.
  - $\mathsf{XOR}(B_1, \ldots, B_n)$ *denotes the block with vertex set* $\bigcup_{i=0}^{n} V(B_i) \cup \{\mathsf{xsplit}, \mathsf{xjoin}\}$ *and edge set* $\bigcup_{i=0}^{n}(E(B_i) \cup \{(\mathsf{xsplit}, b(B_i)), (f(B_i), \mathsf{xjoin}) : 1 \leq i \leq n\})$ *where* xsplit *and* xjoin *respectively denote an* xor_split *coordinator and an* xor_join *coordinator, respectively. The initial element of* $\mathsf{XOR}(B_1, \ldots, B_n)$ *is* xsplit *and its final element is* xjoin.
  - $\mathsf{AND}(B_1, \ldots, B_n)$ *denotes the block with vertex set* $\bigcup_{i=0}^{n} V(B_i) \cup \{\mathsf{asplit}, \mathsf{ajoin}\}$ *and edge set* $\bigcup_{i=0}^{n}(E(B_i) \cup \{(\mathsf{asplit}, b(B_i)), (f(B_i), \mathsf{ajoin}) : 1 \leq i \leq n\})$ *where* asplit *and* ajoin *denote an* and_split *and an* and_join *coordinator, respectively. The initial element of* $\mathsf{AND}(B_1, \ldots, B_n)$ *is* asplit *and its final element is* ajoin.

By enclosing a process block as defined in Definition 1 along with a start and end task in a sequence block, we obtain a *structured process model*. The effects of executing the tasks contained in a business process model are described using annotations as shown in Definition 2.

Definition 2 (Annotated process). *Let P be a structured process and let T be the set of tasks contained in P. An annotated process is a pair: (P, ann), where* ann *is a function associating to each task in T a consistent set of literals:* $\mathsf{ann} : T \mapsto 2^{\mathscr{L}}$. *The function* ann *is constrained to the consistent literals sets in* $2^{\mathscr{L}}$.

The update between the states of a process during its execution is inspired by the AGM[2] belief revision operator [2] and is used in the context of business processes to define the transitions between states, which in turn are used to define the *traces*.

Definition 3 (State update). *Given two consistent sets of literals L_1 and L_2, representing the process state and the annotation of a task being executed, the update of L_1 with L_2, denoted by $L_1 \oplus L_2$ is a set of literals defined as follows:*

$$L_1 \oplus L_2 = L_1 \setminus \{\neg l \mid l \in L_2\} \cup L_2$$

Definition 4 (Executions and Traces). *Given a structured process model identified by a process block B, the set of its executions, written $\Sigma(B) = \{\varepsilon | \varepsilon$ is a sequence and is an execution of $B\}$. The function $\Sigma(B)$ is defined as follows:*

1. *If B is a task t, then $\Sigma(B) = \{(\{t\}, \emptyset)\}$*
2. *if B is a composite block with sub-blocks $B_1, \ldots, B_n$:*
   (a) *If $B = \mathsf{SEQ}(B_1, \ldots, B_n)$, then $\Sigma(B) = \{\varepsilon_1 +_{\mathscr{E}} \cdots +_{\mathscr{E}} \varepsilon_n | \varepsilon_i \in \Sigma(B_i)\}$ and $+_{\mathscr{E}}$ the operator concatenating two executions.*
   (b) *If $B = \mathsf{XOR}(B_1, \ldots, B_n)$, then $\Sigma(B) = \Sigma(B_1) \cup \cdots \cup \Sigma(B_n)$*
   (c) *If $B = \mathsf{AND}(B_1, \ldots, B_n)$, then $\Sigma(B) = \{$the union of the interleavings of: ε_1, $\ldots$, $\varepsilon_n | \varepsilon_i \in \Sigma(B_i)\}$*

[2] The operator is named after the initials of the authors introducing it: Alchourrón, Gärdenfors, and Makinson.

Given an annotated process (B, ann) *and an execution* $\varepsilon = (t_1, \ldots, t_n)$ *such that* $\varepsilon \in \Sigma(B)$*, a trace* θ *is a finite sequence of states:* $(\sigma_1, \ldots, \sigma_n)$*. Each state of* $\sigma_i \in \theta$ *is a pair:* (t_i, L_i) *capturing what holds after the execution of a task* t_i*, expressed by a set of literals* L_i*. A set* L_i *is constructed as follows:*

1. $L_0 = \emptyset$
2. $L_{i+1} = L_i \oplus \mathsf{ann}(t_{i+1})$*, for* $1 \leq i < n$*.*

To denote the set of possible traces resulting from a process model (B, ann)*, we use* $\Theta(B, \mathsf{ann})$*.*

Example 1. Annotated Process Model. Fig. 1 shows a structured process containing four tasks labelled t_1, t_2, t_3 and t_4 and their annotations. The process contains an AND block followed by a task and an XOR block nested within the AND block. The annotations indicate what has to hold after a task is executed. If t_1 is executed, then the literal a has to hold in that state of the process.

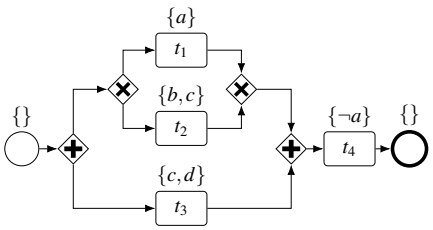

Fig. 1. An annotated process

$\Sigma(B)$	$\Theta(B, \mathsf{ann})$
$(\mathsf{start}, t_1, t_3, t_4, \mathsf{end})$	$((\mathsf{start}, \emptyset), (t_1, \{a\}), (t_3, \{a, c, d\}), (t_4, \{\neg a, c, d\}), (\mathsf{end}, \{\neg a, c, d\}))$
$(\mathsf{start}, t_2, t_3, t_4 \mathsf{end})$	$((\mathsf{start}, \emptyset), (t_2, \{b, c\}), (t_3, \{b, c, d\}), (t_4, \{\neg a, b, c, d\}), (\mathsf{end}, \{\neg a, b, c, d\}))$
$(\mathsf{start}, t_3, t_1, t_4 \mathsf{end})$	$((\mathsf{start}, \emptyset), (t_3, \{c, d\}), (t_1, \{a, c, d\}), (t_4, \{\neg a, c, d\}), (\mathsf{end}, \{\neg a, c, d\}))$
$(\mathsf{start}, t_3, t_2, t_4 \mathsf{end})$	$((\mathsf{start}, \emptyset), (t_3, \{c, d\}), (t_2, \{b, c, d\}), (t_4.\{\neg a, b, c, d\}), (\mathsf{end}, \{\neg a, b, c, d\}))$

Table 1. Executions and Traces of the annotated process in Fig. 1.

2.2 Regulatory Framework

We hereby introduce the regulatory framework through the use of obligations, which are the conditions that a process model needs to fulfil in order to be considered compliant. As such, we use a subset of Process Compliance Logic (PCL), introduced by Governatori and Rotolo [9]. Obligations can be either *global* or *local*. A global obligation is in force for the entire duration of an execution, while the in force interval of a local obligation is determined by its trigger and deadline conditions. Additionally, an obligation can be either an achievement or a maintenance obligation and it determines how such an obligation is fulfilled by an execution when in force.

Definition 5 (Global and Local Obligations).

Global *A global obligation* $\mathscr{O}^o\langle c \rangle$*, where* $o \in \{a, m\}$ *represents whether the obligation is achievement or maintenance. The element c represents the condition of the obligation.*

Local *A local obligation* $\mathscr{O}^o\langle c, t, d \rangle$*, where* $o \in \{a, m\}$ *represents whether the obligation is achievement or maintenance. The element c represents the condition of the obligation, the element t the trigger, and the element d the deadline.*

While the in force interval of a global obligation spans the entire duration of a trace, the in force interval of a local obligation is determined as a sub-trace where the first state of such a sub-trace satisfies the trigger, and the last state satisfies the deadline. Generally the trigger, deadline and condition of an obligation are defined as propositional formulae. Assuming the literals from $\mathscr{L}$ contained in a state to be true, then a propositional formula can then be evaluated when that state implies it.

Evaluating the Obligations. Whether an in force obligation is fulfilled or violated is determined by the states of the trace included in the in force interval of the obligation. Moreover, whether an in force obligation is fulfilled depends on the type of an obligation, as described in Definition 6.

Definition 6 (Achievement and Maintenance Obligations).

Achievement *If this type of obligation is in force in an interval, then the fulfilment condition specified by the regulation must be satisfied by the execution in at least one point in the interval before the deadline is satisfied. If this is the case then the obligation in force is considered to be satisfied, otherwise it is violated.*

Maintenance *If this type of obligation is in force in an interval then the fulfilment condition must be satisfied* continuously *in all points of the interval until the deadline is satisfied. Again, if this is the case then the obligation in force is then satisfied, otherwise it is violated.*

Process Compliance. The procedure of proving whether a process is compliant with a regulatory framework can return different answers. A process is said to be *fully compliant* if every trace of the process is compliant with the regulatory framework. A process is *partially compliant* if there exists at least one trace that is compliant with the regulatory framework, and *n*ot compliant if there is no trace complying with the framework.

Definition 7 (Process Compliance). *Given a process* (P, ann) *and a regulatory framework composed by a set of obligations* $\mathbb{O}$*, the compliance of* (P, ann) *with respect to* $\mathbb{O}$ *is determined as follows:*

- ***Full compliance*** $(P, \mathsf{ann}) \vdash^{F} \mathbb{O}$ *if and only if* $\forall \theta \in \Theta(P, \mathsf{ann}), \theta \vdash \mathbb{O}$.
- ***Partial compliance*** $(P, \mathsf{ann}) \vdash^{P} \mathbb{O}$ *if and only if* $\exists \theta \in \Theta(P, \mathsf{ann}), \theta \vdash \mathbb{O}$.
- ***Not compliant*** $(P, \mathsf{ann}) \nvdash \mathbb{O}$ *if and only if* $\neg\exists \theta \in \Theta(P, \mathsf{ann}), \theta \vdash \mathbb{O}$.

Note that we consider a trace to be compliant with a regulatory framework if it satisfies every obligation belonging to the set composing the framework.

3 Existing Complexity Results

In this section, we introduce some of the existing computational complexity results related variants of the problem discussed in this paper. In particular, we use such results as a starting point from which we analyse the elements bringing the computational complexity of the problem into the class of **NP** problems.

3.1 Problem Acronyms

Before discussing the existing results, we first introduce a compact system to refer to different variants of the problem through acronyms representing their properties.

Definition 8 (Compact Acronyms). *The variants of the problem we refer to in this paper constantly aim to check regulatory compliance of a structured process model. The acronym system refers to the properties of the obligations being checked against the process model.*

1/n *Whether the structured process is checked against a single (**1**) or a set of (**n**) obligations.*

G/L *Whether the* in force *interval of the obligations is **G**lobal, meaning that it spans the entirety of an execution of the model, or it is **L**ocal, meaning that the in force interval is determined by the trigger and deadline elements of an obligation.*

-/+ *Whether the elements of the obligation being checked on the structured process model are composed by literals (**-**), or by propositional formulae (**+**).*

For instance, the variant **1G-** consists of verifying whether a structured process model is compliant with a single obligation, whose condition is expressed as a propositional literal and its in force interval spans the entire execution of the process model. Note that in the binary properties of the problems considered in this paper, the leftmost, i.e. **1** in **1/n** represents a subset of the right side. Intuitively, the case on the right side is at least as complex as the left case. For instance, a solution for a problem including a set of regulations requires also to solve the case where the set of regulations is composed of exactly 1 regulation.

3.2 Complexity Results

Table 2 outlines the existing complexity results. Note that, even if Colombo Tosatto et al. provide a reduction showing how the Hamiltonian Path problem can be reduced to **nL-**, the same reduction can also be used to show that the more expressive **nL+** is in **NP**-complete.

Problem	Source	Compliance	Complexity Class
1G-	Colombo Tosatto et al. [23]	Full and Partial	**P**
nL-	Colombo Tosatto et al. [22]	Partial	**NP**-complete
1L+	Colombo Tosatto et al. [22]	Full	co**NP**-complete
nL+	Colombo Tosatto et al. [22]	Partial	**NP**-complete

Table 2. Regulatory Compliance Complexity

From the existing results, it can be noticed that the computational complexity of the variants of the problem jumps from **P** to **NP** as soon as the obligations become **L**ocal, and either the number of obligations is more than one or the elements describing them can be represented by propositional formulae.

Partial Compliance for 1L+ is in NP-complete. Colombo Tosatto et al. [22] show that proving full compliance in **1L+** is co**NP**-complete by providing a reduction from the *tautology* problem. We extend the existing result by showing that verifying whether a structured process model is *partially compliant* with a single local obligation whose elements are expressed using formulae is in **NP**-complete.

We show that proving partial compliance in **1L+** is in **NP**-complete by exploiting the relation between *Tautology* and *Satisfiability* problems. The relation is that in a tautology problem, it is required that every interpretation satisfies the formula, while in a satisfiability problem it is required that there exists one. This relation is the same as the one between *full* and *partial* compliance, where the former checks whether every execution satisfies the regulatory framework, while the latter checks whether there is one. Thus, through the relation mentioned above, and by reusing some elements in the reduction used for the proof showing that proving full compliance of **1L+** is in co**NP**-complete, we prove that checking whether **1L+** is partial compliant is in **NP**-complete.

4 Further Complexity Results

Starting from the existing computational complexity results, we analyse their relations and provide some additional results and insights on the matter.

4.1 From Global to Local Obligations

From Table 2 it might seem that the feature of the problems increasing the computational complexity to the **NP** class is changing the regulations from being global to local, as this very feature is the common one between the variants **nL-** and **1L+**. However, we argue that such a property is not the main cause of the computational complexity jump from **P** to **NP**.

Considering now the variant **nG-**, where each obligation contains a single literal in its condition, we can reduce it to **2G+**, which is a special case of **nG+**, where the set of regulatory requirements contain exactly 2 regulations. This is effectively the simplest case of **nG+**.

Reduction 1 *Consider a set of regulations $\mathbb{R}$, where each regulation r_i contains a fulfilment condition c_i. Consider now the complementary subsets of $\mathbb{R}$: $\mathbb{R}^a$ and $\mathbb{R}^m$, which respectively contain the obligations of type achievement and of type maintenance originally in $\mathbb{R}$.*

*The variant **nG-** can be reduced to **nG+** with an obligation set containing 2 obligations, by collapsing each obligation belonging to one of the subsets into a single one. The obligation resulting from the reduction contains a condition constructed as follows: $\bigwedge_{i=0}^{n} c_i$. This results in a problem containing two obligations, one achievement and one maintenance, which are in force for the entire execution of the process model and whose conditions are represented by conjunctions of literals.*

Proof. The correctness of the reduction follows directly from Definition 6.

The reduction allows us to claim that from a computational complexity perspective **nG-** $\leq$ **nG+**, when proving partial compliance of business process models. This result does not look too surprising initially as the second problem is clearly more complex

than the first, due to stepping from **-** to **+**, while seemingly maintaining the other properties. However, for the sake of precision, the reduction does reduce **nG-** to **2G+**, as the reduced problem contains exactly two obligations, one for each of the possible types. In the next step of the analysis, we focus on the computational complexity of **1G+**.

Partial Compliance of 1G+ is in NP-complete. We prove now that **1G+**, aiming at proving whether a business process model is partially compliant with a single global obligation, whose condition is represented by a formula, is in **NP**-complete. We use a similar approach to the one used to prove the computational complexity class of **1L+**. The difference between the problems is that the current one is limited to use a global obligation.

Despite the differences, we can still reduce *satisfiability* to **1G+**, showing that it indeed belongs to the **NP**-complete complexity class. Considering **1G-** as a starting variant, this also shows that stepping from the simplest elements of the regulation being verified (whose elements are composed of literals) to the more complex variant (where formulae are allowed) brings it into **NP**. Thus, despite the first impression that moving from global to local obligations could have been the main cause of the computational complexity increase, we have shown here that the expressivity of the elements of the obligations indeed contributes to the computational complexity.

Partial Compliance of nG- is in NP-complete. We prove now that **nG-** (aiming at proving whether a business process model is partially compliant with a set of global obligations, whose condition is represented by a single literal) is in **NP**-complete.

When considering a satisfiability problem like 3-SAT, consisting of deciding whether a formula composed of a conjunction of clauses, where each clause is composed of a disjunction of at least 3 literals, is satisfiable. This problem is computationally hard, as illustrated by Karp [14]. Differently, if the number of disjoint literals is lower than 3, then the problem is solvable in polynomial time, as Krom illustrates [17].

Intuitively, when considering **nG-** and knowing that **1G-** is in **P**, then similarly to a satisfiability problem where multiple clauses are involved in the formula being verified mirrored in a compliance problem by the number of obligations, the computational complexity of proving partial compliance in **nG-** should also be in **NP**. We prove that **nG-** is in **NP** by reducing *3-SAT* to it.

4.2 Computational Complexity Recap

Table 3 provides an overview of the computational complexity results introduced in this paper for the problem of proving partial compliance of business process models with respect to a set of obligations with varying features. We can immediately see that the variants of the problem analysed in this paper have been proven to be **NP**-complete.

Problem	Compliance	Complexity Class
nG-	Partial	**NP**-complete
1G+	Partial	**NP**-complete
1L+	Partial	**NP**-complete

Table 3. Additional Regulatory Compliance Complexity

A variant having difficult features that are a superset of another, is at least as difficult as the other. For instance, the variant **nG+** is at least as difficult as both **nG-** or **1G+**. Furthermore, following the same reasoning, it can also be said that **nL+** is at least as difficult as **nG+**. Thus the following computational complexity relations are true: **nG-** $\leq$ **nG+** $\leq$ **nL+** and **1G+** $\leq$ **nG+** $\leq$ **nL+**.

Consider now Table 2 and Table 3. Note that we are missing a computational complexity result for **nG+**. However, it can be seen in the computational complexity relations that **nG+** is at most as difficult as **1G+** and at most as difficult as **nL+**. Knowing that both **1G+** and **nL+** are in **NP**-complete, we can conclude that **nG+** is also in **NP**-complete.

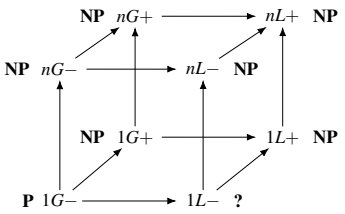

Fig. 2. Partial Compliance Complexity Lattice.

Additionally, aggregating the existing results from Table 2 with the newly obtained results in Table 3, we obtain the computational complexity map for proving partial compliance shown in Figure 2. Finally, note that Figure 2 assigns a complexity class to each problem with the exception of **1L-**. We conclude this section by proposing a conjecture regarding the computational complexity of **1L-**.

*Conjecture 1 (**1L-** is in **P**).* We currently have no information about the computational complexity of **1L-**. That is, we cannot infer its belonging to a computational complexity class in a similar way as for **nG+**, as in this case the simpler variant (**1G-**) is in **P**.

Our conjecture is that the computational complexity of **1L-** is in **P**. We have proven that moving from **-** to **+**, or from **1** to **n**, definitely brings the complexity into **NP**. In general, solutions tackling such variants have to explore the entire set of possible executions in the worst case scenarios, which precludes efficient solutions. Despite moving from **G** to **L** seems to definitely increase the complexity, we strongly believe that it does not influence the computational complexity of the problem enough to move it into **NP**, and polynomial solutions are still possible.

5 Existing Approaches and Their Complexity

Compliance checking in itself is a generic term, and it is used in different ways in various approaches and the type of rules and regulations that are supported (along with the properties supported) differ for each approach. As a result, each approach may correspond to a different problem class as defined in this paper.

For example, approaches such as [3,18] that are merely checking the control-flow of a process using temporal logic or other logic are typically **nG-**. However, control-flow constraints may be conditional and apply therefore only locally, e.g. when using CTL* such as in [10]. In these cases, the enforcing part of the rules can be conditional and the problem belongs to the variant **nL-**. Similarly, when variables (i.e. literals) are included in the approach in temporal logic [5], the problem can immediately be classified as the variant **nL-** as well. Approaches based on classical logic where propositional formulae are supported in addition to literals [16,7,20,8,13,11] can be classified as **nL+**.

As a result of the inherent computational complexity of such approaches, several methods describe a state-space reduction to limit state explosion in concurrent processes. However, these approaches either generate large amounts of overhead (e.g.

[19]), or lose information on concurrency and local activity orders in concurrent branches due to linearisation (e.g., [4,21,6,12]). The reduction approach described in [10] does reduce the state space while preserving all concurrency information, but provides only a practical optimisation and the theoretical complexity (**nL-**) remains the same.

As such, each of the approaches discussed above is **NP**-complete, hence they are bound to be come infeasible, due to the current race towards digitalisation and automatisation.

6 Conclusion

In this paper, we discussed and analysed the computational complexity of some variants of the problem of proving partial compliance of structured business process models. Despite lacking a computational complexity proof for **1L-**, the results provided along with the classifications of some existing approaches tackling the problem, shows that many solutions belong to the **NP**-complete computational complexity class.

While this was not unexpected, we claim that the current focus on optimisation is bound to lead to many complications in the future, as it is disregarding the worst case scenarios due to the limited size of current practical problems. Digitalisation and automatisation is leading towards bigger and more complex automated processes, which current approaches would miserably fail to solve due to the, largely ignored, computational complexity of the problem. While optimising approaches tailored to current problems is still useful, dealing with the *elephant in the room* becomes more and more crucial, and will soon be impossible to ignore it.

As future work to complete the computational complexity picture of the variants covered by the umbrella of **nL+**, we plan to prove the computational complexity of the problem **1L-**. Moreover, we also plan to switch the computational complexity analysis focus from the regulatory framework to the business process model, and studying how different variants of the business process model (i.e. including loops) influence the computational complexity of the problem.

Acknowledgments

This research is supported by the Science and Industry Endowment Fund.

References

1. Wil M. P. van der Aalst. Verification of workflow nets. In *Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, ICATPN '97, pages 407–426, London, UK, UK, 1997. Springer-Verlag.
2. Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530, 1985.
3. Ahmed Awad, Gero Decker, and Mathias Weske. Efficient compliance checking using bpmn-q and temporal logic. In *International Conference on Business Process Management*, pages 326–341. Springer, 2008.
4. Yongsun Choi and J Leon Zhao. Decomposition-based verification of cyclic workflows. In *Automated Technology for Verification and Analysis*, pages 84–98. Springer, 2005.

5. Amal Elgammal, Oktay Turetken, Willem-Jan van den Heuvel, and Mike Papazoglou. Formalizing and appling compliance patterns for business process compliance. *Software & Systems Modeling*, 15(1):119–146, 2016.
6. Sven Feja, Andreas Speck, and Elke Pulvermüller. Business process verification. In *GI Jahrestagung*, pages 4037–4051, 2009.
7. Aditya Ghose and George Koliadis. Auditing business process compliance. In *ICSOC 2007*, pages 169–180, 2007.
8. Guido Governatori. The regorous approach to process compliance. In *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop*, pages 33–40. IEEE, 2015.
9. Guido Governatori and Antonino Rotolo. Norm compliance in business process modeling. In *Proceedings of the 4th International Web Rule Symposium: Research Based and Industry Focused (RuleML 2010)*, volume 6403 of *LNCS*, pages 194–209. Springer, 2010.
10. Heerko Groefsema, Nick R T P van Beest, and Marco Aiello. A formal model for compliance verification of service compositions. *IEEE Transactions on Services Computing*, 11(3):466–479, 2018.
11. Stephan Haarmann, Kimon Batoulis, and Mathias Weske. Compliance checking for decision-aware process models. In *International Conference on Business Process Management*, pages 494–506. Springer, 2018.
12. Jrg Hoffmann, Ingo Weber, and Guido Governatori. On compliance checking for clausal constraints in annotated process models. *Information Systems Frontiers*, 14(2):155–177, 2012.
13. Conrad Indiono, Walid Fdhila, and Stefanie Rinderle-Ma. Evolution of instance-spanning constraints in process aware information systems. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 298–317. Springer, 2018.
14. Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.
15. Bartek Kiepuszewski, Arthur H. M. ter Hofstede, and Christoph Bussler. On structured workflow modelling. In *Proceedings of the 12th International Conference on Advanced Information Systems Engineering*, CAiSE '00, pages 431–445, London, UK, UK, 2000. Springer-Verlag.
16. David Knuplesch, Linh Thao Ly, Stefanie Rinderle-Ma, Holger Pfeifer, and Peter Dadam. On enabling data-aware compliance checking of business process models. In *International Conference on Conceptual Modeling*, pages 332–346. Springer, 2010.
17. Melven R. Krom. The decision problem for a class of first-order formulas in which all disjunctions are binary. *Mathematical Logic Quarterly*, 13(12):15–20, 1967.
18. Ruopeng Lu, Shazia Sadiq, and Guido Governatori. Compliance aware business process design. In *International Conference on Business Process Management*, pages 120–131. Springer, 2007.
19. Shin Nakajima. Verification of Web service flows with model-checking techniques. In *Proc. Int. Symp. on Cyber Worlds*, pages 378–385, 2002.
20. Shazia Sadiq, Guido Governatori, and Kioumars Namiri. Modeling control objectives for business process compliance. In *International conference on business process management*, pages 149–164. Springer, 2007.
21. Shazia W. Sadiq, Maria E. Orlowska, and Wasim Sadiq. Specification and validation of process constraints for flexible workflows. *Information System*, 30(5):349–378, 2005.
22. Silvano Colombo Tosatto, Guido Governatori, and Pierre Kelsen. Business process regulatory compliance is hard. *IEEE Trans. Services Computing*, 8(6):958–970, 2015.
23. Silvano Colombo Tosatto, Pierre Kelsen, Qin Ma, Marwane El Kharbili, Guido Governatori, and Leendert W. N. van der Torre. Algorithms for tractable compliance problems. *Frontiers Comput. Sci.*, 9(1):55–74, 2015.
24. Wil M. P. van der Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

A Proofs

A.1 Proving NP-completenes in General

Definition 9 (NP-complete). *A decision problem is **NP**-complete if it is in the set of **NP** problems and if every problem in **NP** is polynomial-time many-one reducible to it.*

To prove membership in **NP** of a variant of the problem of proving partial compliance, we show that a process is partially compliant with a set of obligations if and only if there is a certificate whose size is at most polynomial in terms of the length of the input (comprising the business process model and the set of obligations) with which we can check whether it fulfils the regulatory framework in polynomial time. As a certificate we use a trace of the model and we check whether it satisfies the regulatory framework.

NP Membership The following algorithm allows to verify whether a trace of a business process model satisfies a set of obligations in polynomial time with respect to the length of the execution.

Algorithm 1 *Given a set of obligations $\mathbb{O}$ and a trace $\theta = (\sigma_{start}, \sigma_1, \ldots, \sigma_n, \sigma_{end})$ such that $\sigma_{start} = (\mathsf{start}, L_0)$ and $\theta \in \Theta(P, \mathsf{ann})$, the algorithm $A_1(\theta, \mathbb{O})$ is defined as follows:*

Certificate Checking Algorithm

Let $\mathsf{Ob} = \emptyset$ *be the set of in force obligations*
for each σ *in* θ **do**
  for each $\omega = \mathscr{O}^o\langle c,t,d\rangle$ *in* $\mathbb{O}$ **do**
    if $\sigma \models t$ **then**
      $\mathsf{Ob} = \mathsf{Ob} \cup \omega$
    end if
  end for each
  for each $\omega = \mathscr{O}^o\langle c,t,d\rangle$ *in* Ob **do**
    if $o = a$ **then**
      if $\sigma \models c$ **then**
        $\mathsf{Ob} = \mathsf{Ob} \setminus \omega$
      else
        if $\sigma \models d$ **then**
          return $\theta \nvdash \mathbb{O}$
        end if
      end if
    else
      if $t = m$ **then**
        if $\sigma \not\models c$ **then**
          return $\theta \nvdash \mathbb{O}$
        end if
        if $\sigma \models d$ **then**
          $\mathsf{Ob} = \mathsf{Ob} \setminus \omega$
        end if
      end if
    end if
  end for each
end for each

return $\theta \vdash \mathbb{O}$;

*Notice that in case of **G**lobal obligations, whose in force interval spans the entirety of the execution, the set* Ob *is pre-loaded with each obligation of such type.*

Algorithm 1 identifies whether a trace fulfils a set of obligations. If this is the case, then following from Definition 7 it is a sufficient condition to say that the process model is partially compliant with the regulatory framework defined by such obligations.

Complexity: the complexity of checking whether a trace is compliant with the set of obligations using Algorithm 1 is at most polynomial in time $\mathbf{O}(n \times o)$ where n is the number of tasks in the process and o is the number of obligations. Since checking whether the interpretation provided in a state satisfies a formula φ can be done in constant time, it does not affects the complexity of the algorithm. The above asymptotic time bound is at most polynomial in the length of the input (which includes the size of both the process and the set of obligations). Therefore, we can conclude that verifying Partial Compliance is indeed in **NP**.

Note that Algorithm 1 allows to check whether a trace is compliant with a regulatory framework where the class of the problem is **nL+**. With **nL+** being the most generic of the problem classes discussed in the paper, the same algorithm can be used to verify traces for every other problem. This allows to prove **NP**-completeness of various classes of the problems by just showing their **NP**-hardness, as their membership is taken care of by Algorithm 1. In the remainder of the appendix, we prove **NP** hardness of some of the problem classes by reducing existing **NP**-complete problems to the problem of proving partial regulatory compliance of a process.

A.2 NP Hardness of 1L+

We reduce the satisfiability problem to **1L+**.

Definition 10 (Satisfiability Problem). *The satisfiability problem is the problem of determining if there exists an interpretation that satisfies a given propositional formula. In other words, it asks whether there exists an interpretation of the propositions in such a way that the formula evaluates to* `true`*. If this is the case, the formula is satisfiable.*

Reduction Let φ be a propositional formula for which we want to verify whether it is satisfiable or not, as described in Definition 10, and let L be the set of propositions contained in φ.

For each proposition belonging to L, we construct an XOR block containing two tasks, one labelled and containing in its annotation the positive proposition (i.e. l) and the other the negative counterpart (i.e. $\neg l$). Notice that we have not explicitly labeled the task, as they can be distinguished by their annotations. All the XOR blocks constructed from L are then included within a single AND block. This AND block is in turn followed by a task labelled "test" and containing a single literal in its annotation: l_{test}. The sequence containing the AND block and the task *test* is then enclosed within a start and an end, composing the annotated business process model (P, ann), as graphically represented in Figure 3.

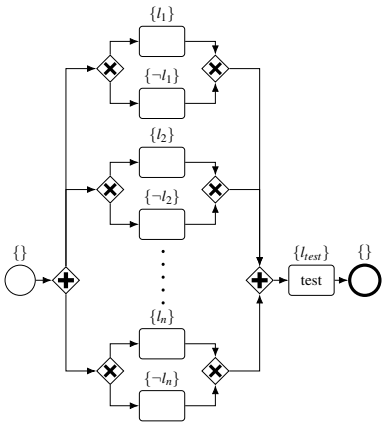

Fig. 3. Process for **1L+** reduction.

The set of obligations, to which the constructed business process has to be verified to be fully compliant with, is composed of a single obligation constructed as follows from the propositional formula φ: $\langle \mathscr{O}^a l_{test}, \varphi, \bot \rangle$. We claim that there exists a trace $\theta \in \Theta(P, \mathsf{ann})$ such that $\theta \vdash \mathbb{O}$ if and only if φ is satisfiable.

Reduction Complexity: The process P and the obligation $\langle \mathscr{O}^a l_{test}, \varphi, \bot \rangle$ can be constructed in time proportional to $|L| + |\varphi|$ where $|\varphi|$ denotes the length of formula φ. Since $|L| \leq |\varphi|$ by construction, the time is at most polynomial in the length of the formula φ.

Correctness Here we prove the soundness ($(P, \mathsf{ann}) \stackrel{\mathrm{P}}{\vdash} \mathbb{O} \Rightarrow \varphi$ is satisfiable) and the completeness (φ is satisfiable $\Rightarrow (P, \mathsf{ann}) \stackrel{\mathrm{P}}{\vdash} \mathbb{O}$) of our reduction.

Proof. **Soundness:** $(P, \mathsf{ann}) \stackrel{\mathrm{P}}{\vdash} \mathbb{O} \Rightarrow \varphi$ is satisfiable.

From the hypothesis and Definition 7, we know that there exists a trace of the business process model (P, ann) that fulfils the obligation in $\mathbb{O}$. Following from the construction of the reduction, we know that the only obligation belonging to $\mathbb{O}$ is $\langle \mathscr{O}^a l_{test}, \varphi, \bot \rangle$.

From Definition 4 and the construction of the reduction we know that each trace of P contains the task l_{test}. Therefore, according to Definition 6, in order for the obligation $\langle \mathscr{O}^a l_{test}, \varphi, \bot \rangle$ to be fulfilled each trace contains a state following the one where l_{test} appears.

From Definition 4 and the construction of the reduction, in particular how (P, ann) is constructed, we have that in the only state following the one where l_{test} appears the first time, the set of literals associated to that state corresponds to an interpretation of the propositions contained in φ. Moreover, again from the construction of the reduction, we know that in all the traces of (P, ann), all the possible combinations of interpreting the propositions belonging to φ are considered.

Therefore, since the obligation $\langle \mathscr{O}^a l_{test}, \varphi, \bot \rangle$ is fulfilled by at least a trace and each trace corresponds to an interpretation, it follows from Definition 10 that φ is indeed satisfiable by the interpretation corresponding to the execution satisfying the obligation.

Proof. **Completeness:** φ is satisfiable $\Rightarrow (P, \mathsf{ann}) \vdash^{\mathrm{P}} \mathbb{O}$.

From the construction of the reduction we know that the condition of the only obligation contained in $\mathbb{O}$ is constituted by φ. However, from the hypothesis we know that φ is satisfiable. Hence, according to Definition 6 such obligation is fulfilled by at least an interpretation of its proposition. Therefore, as by construction the process contains every possible interpretation for the formulae being analysed, and from Definition 7 it follows that if φ is satisfiable, then the compliance problem constructed using the reduction results in partial compliance, as one execution must fulfil the obligation.

A.3 NP Hardness of 1G+

We reduce the satisfiability problem to proving partial regulatory compliance for **1G+**.

Reduction Let φ be a propositional formula for which we want to verify whether it is satisfiable, and let L be the set of propositions in φ. For each proposition l belonging to L, we construct an XOR block containing two tasks, one labelled and containing in its annotation the positive proposition (i.e. l) and the other the negative counterpart (i.e. $\neg l$). All the XOR blocks constructed from L are then included within a single AND block.

The set of obligations, to which the constructed process has to be verified to be fully compliant with, is composed of a single obligation constructed as follows from the propositional formula φ: $\langle \mathscr{O}^a \varphi \rangle$. We claim that there exists a trace $\theta \in \Theta(P, \mathsf{ann})$ such that $\theta \vdash \mathbb{O}$ if and only if φ is satisfiable.

Note that differently from the construction shown in Figure 3, we do not need to include a *test* task to trigger the verification, as the obligation being global means that it is always in force and its condition is required to be achieved by the end of the execution.

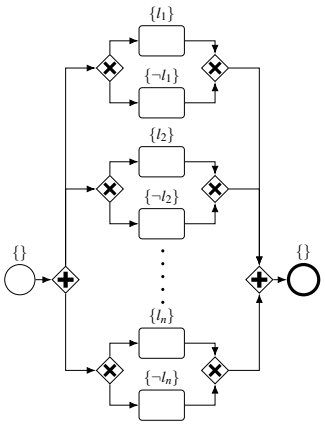

Fig. 4. Process for **1G+** reduction.

Reduction Complexity: As the reduction is similar to the one used for **1L+**, the translation complexity is the same, which is in time polynomial with respect to the size of the formula φ.

Correctness Here we prove the soundness ($(P, \mathsf{ann}) \vdash^{\mathrm{P}} \mathbb{O} \Rightarrow \varphi$ is satisfiable) and the completeness (φ is satisfiable $\Rightarrow (P, \mathsf{ann}) \vdash^{\mathrm{P}} \mathbb{O}$) of our reduction.

Proof. **Soundness:** $(P, \mathsf{ann}) \vdash^{\mathrm{P}} \mathbb{O} \Rightarrow \varphi$ is satisfiable.

From the hypothesis and Definition 7, we know that there exists a trace of the business process model (P, ann) that fulfils the obligation in $\mathbb{O}$. Following from the construction of the reduction we know that the only obligation belonging to $\mathbb{O}$ is $\langle \mathscr{O}^a \varphi \rangle$.

From Definition 4 and the construction of the reduction we know that each trace of P contains a task related to the truth value for each literal appearing in φ. Moreover, again from the construction of the reduction, we know that in all the traces of (P, ann) all the possible combinations of interpreting the propositions belonging to φ are considered.

Therefore, since the obligation $\langle \mathscr{O}^a \varphi \rangle$ is fulfilled by at least a trace and each trace corresponds to an interpretation, it follows from Definition 10 that φ is indeed satisfiable by the interpretation corresponding to the execution satisfying the obligation.

Proof. **Completeness:** φ is satisfiable $\Rightarrow (P, \mathsf{ann}) \vdash^{\mathrm{P}} \mathbb{O}$.

From the construction of the reduction we know that the condition of the only obligation contained in $\mathbb{O}$ is constituted by φ. However, from the hypothesis we know that φ is satisfiable. Hence, according to Definition 6, such obligation is fulfilled by at least an interpretation of its proposition. Therefore, as by construction the process contains every possible interpretation for the formulae being analysed, and from Definition 7, it follows that if φ is satisfiable, then the compliance problem constructed using the reduction results in partial compliance, as one execution must fulfil the obligation.

A.4 NP Hardness of nG-

We reduce the *3-SAT* problem to proving partial regulatory compliance for **nG-**.

Definition 11 (3-SAT). *A propositional formula is in 3-CNF (Conjunctive Normal Form) if it is of the form $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_k$ where each α_i is a disjunction of three or less literals. A propositional formula is in 3-SAT if it is in 3-CNF form and is also satisfiable.*

Reduction Let 3-CNF : $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_k$ be a propositional formula in 3-CNF, as shown in Figure 5, for which we want to verify whether it is satisfiable or not, as described in Definition 11.

For each proposition l_i belonging to 3-CNF, we construct an XOR block containing two tasks, one labelled and containing in its annotation the positive proposition (i.e. a literal l_i) and the other the negative counterpart (i.e. its negation $\neg l_i$). All the XOR blocks constructed from L are then included within a single AND block. This results in the same process model construction used in the computational complexity proof for **1G+** as graphically represented in Figure 4.

$$\underbrace{\{l_1 \vee l_2 \vee l_3\}}_{\alpha_1} \wedge \underbrace{\{\ldots\}}_{\alpha_2} \wedge \ldots \quad \ldots$$

Fig. 5. 3-SAT formula

For each clause α_i in 3-CNF, a global achievement obligation is created and its condition is set to the literal l_{α_i}, referring to the identifier of the clause. For each literal l_j in a clause α_i a *count as* rule is created as follows: $l_j \Rightarrow l_{\alpha_i}$.

Definition 12 (Count As Rule). *Consider a count as rule: $\alpha \Rightarrow \beta$, where α and β are literals. If α is true in the process' state, then the process' state is considered to contain β.*

Notice that the introduction of *count as* in such atomic version (meaning a literal to literal interpretation) does not increase the computational complexity of the problem **nG-**, as such interpretation can be computed in polynomial time while using Algorithm 1 to verify a trace in the reduced problem **nG-**.

Reduction Complexity: The construction of the process model in the reduction is the same as the one used for **1G+**. Differently, the regulatory framework is built by creating a set of obligations and count as rules. Still, constructing the regulatory framework is in time polynomial with respect to the size of 3-CNF. Therefore, the entire reduction is polynomial.

Correctness Here we prove the soundness ($(P,\mathsf{ann}) \vdash^{\mathrm{P}} \mathbb{O} \Rightarrow$ 3-CNF is satisfiable) and the completeness (φ is satisfiable $\Rightarrow (P,\mathsf{ann}) \vdash^{\mathrm{P}} \mathbb{O}$) of our reduction.

Proof. **Soundness:** $(P,\mathsf{ann}) \vdash^{\mathrm{P}} \mathbb{O} \Rightarrow$ 3-CNF is satisfiable.

From the hypothesis and Definition 7 we know that there exists a trace of the business process model (P,ann) that fulfils the obligation in $\mathbb{O}$. Following from the construction of the reduction we know that each obligation in $\mathbb{O}$ is in the form of $\langle \mathscr{O}^a C_i \rangle$.

Moreover, from the construction of the reduction, we know that for each literal in a clause of the 3-CNF formula, a count as rule is created having the literal in its condition and the clause identifier in the conclusion. As each of the obligations in $\mathbb{O}$ is satisfied and the literals used in the condition of the elements of the obligations cannot directly appear in the process' execution state, the condition of at least one of the count as rules associated to a clause must be true in a execution state of the process.

From Definition 4 and the construction of the reduction we know that each trace of P contains a task related to the truth value for each literal appearing in the 3-CNF formula. Moreover, again from the construction of the reduction, we know that in all the traces of (P,ann), all the possible combinations of interpreting the propositions belonging to the 3-CNF formula are considered. Additionally, we also know that: let $\mathscr{I}$ be the final interpretation holding at the end of an execution and let $\mathscr{I}_i$ be an intermediate interpretation, then the following is true $\mathscr{I} \models \mathscr{I}_{i+1} \models \mathscr{I}_i$ for any i.

Therefore, as the interpretation holding at $\mathscr{I}$ is a possible interpretation of the literals in the 3-CNF formula, and the partial interpretation are not in contradiction with the final one, it follows from Definition 11 that the 3-CNF formula is indeed satisfiable by the interpretation corresponding to the execution satisfying the obligations since the obligations $\langle \mathscr{O}^a l_{\alpha_i} \rangle$ are all fulfilled by at least a trace and each trace corresponds to an interpretation, where at least one of the literals in each of the clauses is true.

Proof. **Completeness:** 3-CNF is satisfiable $\Rightarrow (P,\mathsf{ann}) \vdash^{\mathrm{P}} \mathbb{O}$

From the construction of the reduction we know that the obligations contained in $\mathbb{O}$ are constituted by $\langle \mathscr{O}^a l_{\alpha_i} \rangle$. However, from the hypothesis we know that $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_k$ is satisfiable. Hence, according to Definition 6, such obligation is fulfilled by at least an interpretation of its proposition. Moreover, from the construction of the reduction, we know that each literal in a clause forms the condition of a count as rule having the condition of one of the global obligations in its conclusion.

Therefore, as by construction the process contains every possible interpretation for the formulae being analysed, and from the hypothesis we know that $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_k$ is satisfiable, then each clause α_i is satisfied, meaning that one of the disjunct propositions in α_i must be true in the interpretation.

From the construction, and given that there exists an interpretation satisfying the condition of at least a count as rule for each clause, then it follows that each obligation $\langle \mathscr{O}^a l_{\alpha_i} \rangle$ is satisfied. Thus, from Definition 7 it follows that if the 3-CNF is satisfiable then the compliance problem constructed using the reduction results in partial compliance, as one execution must fulfil the obligations.