No Time for Compliance

Guido Governatori and Mustafa Hashmi
NICTA, Brisbane, Australia and
Queensland University of Technology, Brisbane, Australia
Email: {firstname.lastname}@nicta.com.au

***Abstract*—In the past few years several business process compliance frameworks based on temporal logic have been proposed. In this paper we investigate whether the use of temporal logic is suitable for the task at hand: namely to check whether the specifications of a business process are compatible with the formalisation of the norms regulating the business process. We provide an example inspired by real life norms where the use of linear temporal logic produces a result that is not compatible with the legal understanding of the norms in the example.**

I. Introduction

The term compliance can be understood in many different ways. Compliance can be seen as a particular type of verification of a system. Accordingly, business process compliance is the verification of the specifications of a system (modelled using BPM techniques) against a particular normative framework, or more specifically a set of norms. Here, we assume a very broad definition of norms. A norm is a statement by a body/entity (with the authority or power to create, and eventually to enforce, such statements) prescribing or regulating some behaviours such that the non-adherence to the norms potentially leads to some sanctions.

The main idea of business process compliance is to determine if executing a business process according to its specifications does not result in violations of the regulatory frameworks the business activities implemented by the business process have to comply with. This means that in order to verify that a business process is compliant we need two components: (i) the formal specifications of the model of the business process and (ii) the formal representation of the set of norms. The crucial aspect is that both formal representations must be conceptually sound; this boils down to the following issues:

1) correct modelling of processes;
2) correct modelling of norms.

In this paper we assume that the first of these two aspects has been established, and that workflow and business process modelling techniques are suitable for the formal representation of business processes. Similarly, we assume that temporal logics, in particular, Linear Temporal Logic [1] is suitable to model workflows and other approaches to model business processes. Temporal logic is one of the most successful logical formalisms, and it is definitely able to model the sequences of tasks corresponding to a business process, and it is able to efficiently verify large scale industrial applications. These features motivated several researchers to adopt temporal logic as the underlying formalism for their compliance frameworks, see for example [2]–[7]. The analysis and representation of norms in logic and other formal modelling techniques have been thoroughly investigated in the fields of Deontic Logic and Artificial Intelligence and Law. The debate whether temporal logic is suitable for the modelling of norms has a long story from [8] and supporting it to [9] casting serious doubts to the appropriateness of this endeavour. In this paper we are going to argue that the current compliance checking approaches based on Linear Temporal Logic fail on the correct modelling of norms aspect.

In this paper we address the question whether existing compliance frameworks based on (linear) temporal logic are able to provide conceptually sound representations of the regulatory requirements governing a business process. To this means that we are going to investigate whether the techniques, patterns and templates provided by such compliance frameworks to model regulatory requirements in temporal logic are suitable to determine whether a business process (model) complies with a relevant regulatory document (set of norms). To this end, we are going to argue that the current approaches based on (linear) temporal logic suffer from a major limitation for the representation of regulatory requirements. Specifically, we are going to show that either:

1) the techniques and patterns provided by such compliance frameworks are not able to model of notion of permission; or
2) when they do, the frameworks produce paradoxical results.

The consequences of this limitation are, if a regulatory document contains provisions about permissions, then, for

1) the frameworks are not able to determine if a business process complies with the regulatory document;
2) the results they produce can be incorrect: a business process can be assessed as compliant when it is not, and the other way around.

The paper is organised as follows: in the next three sections we provide an outline of the basic foundation of business process modelling (Section II), how to model norms in the context of business processes (Section III), and an outline of Linear Temporal Logic (IV). We then discuss how the frameworks based on temporal logic address the issue of compliance (Section V). The last three sections are dedicated to presenting a compliance scenario inspired by real-life norms (Section VI), how to model the scenario using the techniques

of one of the existing compliance frameworks (Section VII), showing that the representation ends in an outcome that is not compatible with the legal understanding of the scenario, and some final discussion (Section VIII).

II. Business Process Modelling

A business process model is a self-contained, temporal and logical order in which a set of activities are executed to achieve a business goal. Typically a process model describes what needs to be done and when (control flow), who is going to do what (resources), and on what it is working on (data). A typical scenario is that a business process can be executed in many different ways, and a key feature of business process models is that they provide a compact representation of the possible ways in which a process can be executed. A possible execution is called a *trace*, and a trace is then a sequence of tasks to be performed to achieve the business goal of the process. While different languages have been proposed for the modelling of business process, in this paper we ignore the specific aspects of how a process model is represented in a particular language, and we consider a business process model as the set of the possible ways in which the process can be executed; thus we consider a process model as a set of traces.

The performance of each task in a process can change the state of the environment where the process is situated. Thus for each sequence of events or tasks we have a corresponding sequence of states. Compliance is not restricted to what actions are legal or mandatory, but it is also concerned with the effects of the actions, thus to check that a process is compliant we have to check that no illegal actions are performed, all mandatory actions are executed at the right time, and that the process does not result in any illegal states. To account for this aspect [10], [11] proposed to use a bijection that associates each task in a trace to a set of formulas corresponding to the state of the environment after the task has been executed in the particular trace. Notice, that different traces can result in different states, even if the tasks in the traces are the same. In addition, even if the end states are the same, the intermediate states can be different. Accordingly, we extend the notion of trace. First of all, given a process P and the set of traces $\mathcal{T}_P$ corresponding to it[1], we introduce the function

$$State\colon \mathcal{T}_P \times \mathbb{N} \mapsto 2^{\mathcal{L}},$$

where $\mathcal{T}_P$ is the set of traces corresponding to the business process P, and $\mathcal{L}$ is the set of formulas of the language used to model the annotations or the state of the process. Let us illustrate with an example the meaning of the function *State*. Suppose we have the trace $t = \langle A, B, D, E\rangle$, and that $State(t, 3) = \{p, q, r\}$. This means that $\{p, q, r\}$ is the state resulting after executing D in the trace t (D is the third task in t); in other terms p, q and r holds after the execution of D in the given trace t . Notice that a trace uniquely determines the sequence of states obtained by executing the trace. Thus, in what follows we use a trace to refer to a sequence of tasks and the corresponding sequence of states.

III. Modelling Norms

The scope of norms is to regulate the behaviour of their subjects and to define what is legal and what is illegal. Norms typically describe the conditions under which they are applicable and the normative effects they produce when applied. In this paper we concentrate on the deontic effects of norms, i.e., the *obligations*, *prohibitions* and *permissions* that enter in force when norms are triggered.

Let us start by considering the basic definitions for such concepts:[2]

Obligation A situation, an act, or a course of action to which a bearer is legally bound, and if it is not achieved or performed results in a violation.

Prohibition A situation, an act, or a course of action which a bearer should avoid, and if it is achieved results in a violation.

Permission Something is permitted if the obligation to the contrary or its prohibition does not hold.

Obligations and prohibitions are constraints that limit the space of actions of processes; the difference from other types of constraints is that they can be violated, and a violation does not imply an inconsistency within a process with the consequent termination of or impossibility to continue the business process. Furthermore, it is common that violations can be compensated for, and processes with compensated violations are still compliant [14]. Not all violations are compensable, and uncompensated violations means that a process is not compliant. On the other hand, permissions cannot be violated, thus permissions do not play a direct role in compliance. This is one of the reasons why permissions are not considered as first class citizens in a compliance perspective. However, as we are going to show in the rest of the paper, ignoring them can have severe consequences; permissions can be used to determine that there are no obligations or prohibitions to the contrary, or to derive other deontic effects, which then influence the determination if a process is compliant or not. Legal reasoning and legal theory typically assume a strong relationship between obligations and prohibitions: the prohibition of A is the obligation of $\neg A$ (the opposite of A), and then if A is obligatory, then $\neg A$ is forbidden [15]; similarly permission is the lack of obligation to the contrary, thus A is permitted if $\neg A$ is not obligatory. Accordingly, we can restrict our analysis to the notion of *obligation*.

Compliance means to identify whether a process violates or not a set of obligations. Thus, the first step is to determine

[1]We refer to [11], [12] for the full characterisation of the set of traces of a business process model. All we want to remark here is that that there is a one to one correspondence between the tasks in a trace and a (finite) subset of the set of natural numbers $\mathbb{N}$.

[2]Here we consider the definition of such concepts given by the OASIS LegalRuleML working group. The OASIS LegalRuleML glossary is available at http://www.oasis-open.org/apps/org/workgroup/legalruleml/download.php/48435/Glossary.doc. See also [13].

whether and when an obligation is in force. Hence, an important aspect of the study of obligations is to understand the lifespan of an obligation and its implications on the activities carried out in a process. As we have alluded to above norms give the conditions of applicability of obligations. Thus, first, one has to determine when and under what conditions an obligation enters into force. Then, the next question is how long does an obligation hold for, and based on this there are different conditions to fulfil the obligation. In this paper we follow the classification and semantics of obligations proposed in [10], [11]. For the purpose of the paper we restrict our attention to the notions of *achievement obligation*, *maintenance obligation* and *compensable obligation*. The first two notions refer to the life-cycle of an obligation, and the conditions to fulfil or violate the obligation. An obligation remains in force until terminated or removed. Thus the key concept is if to fulfil an obligation we have to obey to it for all instants in the interval in which it is in force, *maintenance obligations*, or whether doing or achieving the content of the obligation at least once is enough to fulfil it, *achievement obligations*. For achievement obligations another aspect to consider is whether the obligation could be fulfilled even before the obligation is actually in force. If this is admitted, then we have a *preemptive obligation*, otherwise the obligation is *non-preemptive*. The notion of compensable obligation is related to the possible effects of violating an obligation, and whether the violation can be compensated for (and the process still be deemed somehow acceptable) or the violation results in an unrecoverable situation.[3]

Definition 1 (Obligation in force): Given a process P, and a trace $t \in \mathcal{T}_P$. We define a function

$$Force\colon \mathcal{T}_P \times \mathbb{N} \mapsto 2^{\mathcal{L}}.$$

The function *Force* associates to each task in a trace a set of literals, where these literals represent the obligations in force for that combination of task and trace. These are among the obligations that the process has to fulfill to comply with a given normative framework. In the rest of the section we are going to give definitions specifying when the process has to fulfill the various obligations (depending on their type) to be deemed compliant.

Definition 2 (Achievement Obligation): Given a process P and a trace $t \in \mathcal{T}_P$, an obligation o is an *achievement obligation* in t if and only if $\exists n, m \in \mathbb{N}, n < m$ such that

1) $o \notin Force(t, n-1)$,
2) $o \notin Force(t, m+1)$, and
3) $\forall k : n \leq k \leq m, o \in Force(t, k)$

An achievement obligation o is *violated* in t if and only if

- o is *preemptive* and $\forall k\colon k \leq m,\ o \notin State(t, k)$;
- o is *non-preemptive* and $\forall k\colon n \leq k \leq m,\ o \notin State(t, k)$.

[3]The definitions in this section are based on [10], [11]. We further refer the reader the same sources for more detailed presentations and concrete examples, for real acts, for all types of obligations.

An achievement obligation is in force in a contiguous set of tasks in a trace. The violation depends on whether we have a preemptive or a non-preemptive obligation. A preemptive obligation o is violated if no state before the last task in which o is in force has o in its annotations; for a non-preemptive obligation the set of states is restricted to those defined by the interval in which the obligation is in force.

Definition 3 (Maintenance Obligation): Given a process P and a trace $t \in \mathcal{T}_P$, an obligation o is a *maintenance obligation* in t if and only if $\exists n, m \in \mathbb{N}$, $n < m$ such that:

1) $o \notin Force(t, n-1)$,
2) $o \notin Force(t, m+1)$, and
3) $\forall k\colon n \leq k \leq m, o \in Force(t, k)$

A maintenance obligation o is *violated* in t if and only if

$$\exists k\colon n \leq k \leq m, o \notin State(t, k).$$

Similarly to an achievement obligation, a maintenance obligation is in force in an interval. The difference is that the obligation has to be complied with for all tasks in the interval, otherwise we have a violation.

The next three definitions are meant to capture the notion of compensation of a violation. The idea is that a compensation is a set of penalties or sanctions imposed on the violator, and fulfilling them makes amend for the violation. The first step is to define what a compensation is. A compensation is a set of obligations in force after a violation of an obligation (Definitions 4 and 5). Since the compensations are obligations themselves they can be violated, and they can be compensable as well, thus we need a recursive definition for the notion of compensated obligation (Definition 6).

Definition 4 (Compensation): A *compensation* is a function $Comp\colon \mathcal{L} \mapsto 2^{\mathcal{L}}$.

Definition 5 (Compensable Obligation): Given a process P and a trace $t \in \mathcal{T}_P$, an obligation o is *compensable* in t if and only if

1) $Comp(o) \neq \emptyset$ and
2) $\forall o' \in Comp(o), \exists n \in \mathbb{N}\colon o' \in Force(t, n)$.

Definition 6 (Compensated Obligation): Given a process P and a trace $t \in \mathcal{T}_P$, an obligation o is *compensated* in t if and only if it is violated and for every $o' \in Comp(o)$ either:

1) o' is not violated in t, or
2) o' is compensated in t.

For a stricter notion, i.e., a compensated compensation does not amend the violation the compensation was meant to compensate, we can simply remove the recursive call, thus removing clause 2) from the above condition.

The set of traces of a given business process describes the behaviour of the process insofar as it provides a description of all possible ways in which the process can be correctly executed. Accordingly, for the purpose of defining what it means for a process to be compliant, we will consider a process as the set of its traces.

Intuitively a process is compliant with a set of norms if it does not violate any norm in the set. Given that, in general, it is possible to perform a business process in many different ways, thus we can have two notions of compliance:

Definition 7: Let $\mathcal{N}$ be a normative system.

1) A process P *fully complies* with $\mathcal{N}$ iff every trace $t \in \mathcal{T}_P$ complies with $\mathcal{N}$.
2) A process P *partially complies* with $\mathcal{N}$ iff there is a trace $t \in \mathcal{T}_P$ that complies with $\mathcal{N}$.

The difference between these two definitions of compliance is that the first case ensures that all possible executions are compliant, i.e., no execution results in a state with (uncompensated) obliations, while the second establishes that it is possible to execute the process without violating the norms. In both cases the definition depends on the notion of "to comply with".

Definition 8: A trace t complies with a normative system $\mathcal{N} = \{n_1, n_2, \dots\}$ iff all norms in $\mathcal{N}$ have not been violated.

We provided various types of norms with their semantics in terms of what constitutes a violation of a norm of that type. The possibility of a norm to be violated is what distinguish norms from other types of constraints. Then, given that violations are possible, one has to consider that violations can be compensated. Is a process where some norms have violated and compensated for compliant? To account for this possibility we introduce the distinction between *strong* and *weak* compliance. Strong compliance corresponds to Definition 8. Weak compliance is defined as follows:

Definition 9: A trace t weakly complies with a normative system $\mathcal{N}$ iff every violated norm has been compensated for.

In the rest of the paper we concentrate on the notion of weakly compliance.

IV. Linear Temporal Logic

Linear Temporal Logic (LTL) [1] is an an extension of classical propositional logic with temporal operators. LTL is equipped with three unary temporal operators:

- $\mathsf{X}\phi$: next ϕ (ϕ holds at the next time);
- $\mathsf{F}\phi$: eventually ϕ (ϕ holds sometimes in the future); and
- $\mathsf{G}\phi$: globally ϕ (ϕ always holds in the future).

In addition LTL has the following binary operators:

- $\phi \,\mathsf{U}\, \psi$: ϕ until ψ (ϕ holds until ψ holds);
- $\phi \,\mathsf{W}\, \psi$: ϕ weak until ψ (ϕ holds until ψ holds and ψ might not hold).

The operators above are related by the following equivalences establishing some interdefinability among them:

- $\mathsf{F}\phi \equiv \top \,\mathsf{U}\, \phi$,
- $\mathsf{G}\phi \equiv \neg\mathsf{F}\neg\phi$,
- $\phi \,\mathsf{W}\, \psi \equiv (\phi \,\mathsf{U}\, \psi) \vee \mathsf{G}\phi$.

The semantics of LTL can be given in terms of transition systems. A *transition system* TS is a structure

$$TS = \langle S, R, v \rangle \tag{1}$$

where

- S is a (non empty) set of states
- $R \subseteq S \times S$ such that $\forall s \in S \exists t \in S \colon (s,t) \in R$
- v is a valuation function $v \colon S \mapsto 2^{Prop}$

where *Prop* is the set of atomic propositions.

Formulas in LTL are evaluated against fullpaths (also called traces or runs). A *fullpath* is a sequence of states in S connected by the transition relation R. Accordingly, $\sigma = s_0, s_1, s_2 \dots$ is a fullpath if and only if $(s_i, s_{i+1}) \in R$. Given a fullpath σ, σ_i denotes the subsequence of σ starting from the i-th element, and $\sigma[i]$ denotes the i-th element of σ.

Equipped with the definitions above, the valuation conditions for the various temporal operators are:

- $TS, \sigma \vDash p$ ($p \in Prop$) iff $p \in v(\sigma[0])$;
- $TS, \sigma \vDash \neg\phi$ iff $TS, \sigma \nvDash \phi$;
- $TS, \sigma \vDash \phi \wedge \psi$ iff $TS, \sigma \vDash \phi$ and $TS, \sigma \vDash \psi$;
- $TS, \sigma \vDash \mathsf{X}\phi$ iff $TS, \sigma_1 \vDash \phi$;
- $TS, \sigma \vDash \phi \,\mathsf{U}\, \psi$ iff $\exists k \colon k \geq 0,\ TS, \sigma_k \vDash \psi$ and $\forall j \colon 0 \leq j < k$, $TS, \sigma_j \vDash \phi$;
- $TS, \sigma \vDash \mathsf{G}\phi$ iff $\forall k \geq 0,\ TS, \sigma_k \vDash \phi$;
- $TS, \sigma \vDash \mathsf{F}\phi$ iff $\exists k \geq 0,\ TS, \sigma_k \vDash \phi$.

A formula ϕ is true in a fullpath σ iff it is true at the first element of the fullpath. Next, we define what it means for a formula ϕ to be true in a state $s \in S$ ($TS, s \vDash \phi$).

$$TS, s \vDash \phi \text{ iff } \forall \sigma \colon \sigma[0] = s,\ TS, \sigma \vDash \phi. \tag{2}$$

As we have just seen, the semantics of LTL is given by a discrete, totally order set of time instants. This structure is isomorphic to a subset of the set of natural numbers, and thus it is isomorphic to a trace of a process. In Section II we introduced the function *State* to populate the state resulting from the execution of the tasks in a trace. It is immediate to see, given a trace t, the correspondence between *State* and the valuation function v. Also, it is easy to model the conditions for definitions of the various types of obligations in LTL. If we ignore the triggering conditions and deadlines, a maintenance obligation can be modelled using G and an achievement obligation using F. The full definition for a maintenance obligation for ϕ can be given by

$$\mathsf{G}(\tau \rightarrow \phi \,\mathsf{U}\, \delta) \tag{3}$$

where τ is a formula corresponding to the condition of activation of the obligation and δ is a formula encoding the deadline for the obligation. Similarly, for an achievement obligation for ϕ we have

$$\mathsf{G}(\tau \rightarrow \neg(\neg\phi \,\mathsf{U}\, \delta)). \tag{4}$$

Compliance then is reduce to problem of determining, given a model encoding a trace of a business process and a set of formulas encoding the relevant norms, whether the formulas are satisfiable by the model.

V. Modelling Business Process Compliance

Automated compliance checking of the legal requirements of business processes is highly desirable. These requirements are written in natural language and must be translated into a machine-readable format for automated verification. Generally formal languages, e.g., event-calculus, temporal logic, deontic logic etc., providing the reasoning support are used to translate the legal requirements. However, these languages are difficult to understand due to their complexity in terms of usability and comprehensibility especially for non-technical users, e.g., process analysts and compliance experts. Thus, usability of the formal languages is one of the main concerns for non-technical users who possess less knowledge of these languages [3]. To address the usability concern of the formal languages, researchers proposed to embed the formulas in a formal language translating the compliance requirements into easy to understand visual patterns or graphs. This lead the emergence of many graph/patterns based compliance verification approaches in business process compliance domain such as BPMN-Q [4], COMPAS [2], [3], DECLARE [7] etc.

In the rest of the section we focus on COMPAS and its underlying Compliance Requirement Language (CRL) since it includes most of the patterns used by other frameworks as well additional patterns meant to represent features specific of normative reasoning, such as rule exception and compensations of violations. However, the analysis can be extended to other compliance frameworks based on temporal logics.

COMPAS is a compliance governance framework which provides all-round compliance support for SOA based systems. The framework is grounded on LTL based graphical patterns representing different types of compliance rules, which are than grouped into three distinct categories of patterns: *atomic patterns*, *resources patterns*, and *composite patterns*. The atomic patterns are based on the Dwyer's property specification patterns [16], and categorised into *occurrence* and *ordering* patterns. These patterns are as follow [3]:

- *isAbsent* (ϕ *isAbsent*): indicates that ϕ should not exist throughout the process model.
  LTL Formula: $\mathsf{G}\neg\phi$
- *Exists* (ϕ *Exists*): represents that ϕ should occur at least once within the process model.
  LTL Formula: $\mathsf{F}\phi$
- *Bounded-Exists* (ϕ *BoundedExist* $\leq 2^*$ *with bound* ≤ 2): shows that ϕ must occur at most 2 times within the process.
  LTL Formula: $\neg\phi\ \mathsf{W}\ (\phi\ \mathsf{W}\ (\neg\phi\ \mathsf{W}\ (\phi\ \mathsf{W}\ \neg\mathsf{F}\phi)))$
- ϕ *BoundedExist* $\geq 2^*$ *with bound* ≥ 2: shows that ϕ must occur at least 2 times within the process.
  LTL Formula: $\neg\phi\ \mathsf{W}\ (\phi\ \mathsf{W}\ (\neg\phi\ \mathsf{W}\ \phi))$
- *isUniversal* (ϕ *isUniversal*): indicates that ϕ should always be true throughout the process.
  LTL Formula: $\mathsf{G}\phi$
- *Precedes* (ϕ *Precedes* ψ): meaning that ψ is always preceded by ϕ.
  LTL Formula: $\neg\psi\ \mathsf{W}\ \phi$
- *Chain-Precedes* (ϕ *Precedes* (σ, τ)): meaning that a sequence of σ, τ must be preceded by ϕ .
  LTL Formula: $\mathsf{F}(\sigma \wedge \mathsf{XF}\tau) \rightarrow (\neg\sigma\ \mathsf{U}\ \phi)$
- (σ, τ) *Precedes* ϕ: meaning that ϕ must be preceded by a sequence of σ, τ.
  LTL Formula: $\mathsf{F}(\phi) \rightarrow (\neg\phi\ \mathsf{U}\ (\sigma \wedge \neg\phi \wedge \mathsf{X}(\neg\phi\ \mathsf{U}\ \tau)))$
- *LeadsTo* (ϕ *LeadsTo* ψ): indicates ϕ must always be followed by ψ.
  LTL Formula: $\mathsf{G}(\phi \rightarrow \mathsf{F}\psi)$
- *Chain-LeadsTo* (ϕ *LeadsTo* (σ, τ)): shows that ϕ must be followed by a sequence of σ, τ.
  LTL Formula: $\mathsf{G}(\phi \rightarrow \mathsf{F}(\sigma \wedge \mathsf{XF}\tau))$
- (σ, τ) *LeadsTo* ϕ: indicates that a sequence of σ, τ must be followed ϕ.
  LTL Formula: $\mathsf{G}(\sigma \wedge \mathsf{XF}(\tau) \rightarrow \mathsf{X}(\mathsf{F}(\tau \wedge \mathsf{F}\phi)))$
- *DirectlyFollowedBy* (ϕ *DirectlyFollowedBy* ψ): shows that Requires ϕ to be followed by ψ.
  LTLFormula: $\mathsf{G}(\phi \rightarrow \mathsf{X}\psi)$

In addition to the atomic patterns CRL defines the following *composite patterns* [3]:

- *CoExists* (ϕ CoExists ψ): meaning that the presence of ϕ mandates that ψ is also present
  LTL Formula $\mathsf{F}\phi \rightarrow \mathsf{F}\psi$
- *CoAbsence* (ϕ CoAbsence ψ): shows the absence of ϕ mandates that ψ is also absence
  LTL Formula $\mathsf{G}\neg\phi \rightarrow \mathsf{G}\neg\psi$
- *Exclusive* (ϕ Exclusive ψ): suggests presence of ϕ mandates the absence of ψ. And the presence of ψ mandates the absence of ϕ
  LTL Formula $(\mathsf{F}\phi \rightarrow \mathsf{G}\neg\psi) \wedge (\mathsf{F}\psi \rightarrow \mathsf{G}\neg\phi)$
- *Substitute* (ψ Substitute ϕ): indicates ψ substitutes the absence of ϕ
  LTL Formula $\mathsf{G}\neg\phi \rightarrow \mathsf{F}\psi$
- *Corequisite* (ψ Corequisite ψ): ϕ and ψ should either exist together or to be absent together
  LTL Formula $(\mathsf{F}\phi \rightarrow \mathsf{F}\psi) \wedge (\mathsf{F}\psi \rightarrow \mathsf{F}\phi)$
- *MutexChoice* (ψ MutexChoice ψ): meaning that either ϕ or ψ exists but not any of them or both of them
  LTL Formula $(\mathsf{F}\phi \wedge \mathsf{G}\neg\psi) \vee (\mathsf{F}\psi \wedge \mathsf{G}\neg\phi)$

These patterns are mapped into LTL formulas (see the description above) enabling the translation of CRL expressions into a set of LTL formulas. Essentially, these patterns are used to represent different types of obligations e.g., achievement, prohibitions etc. In addition, to model contrary-to-duty obligations (compensations) and non-monotonic requirements (exceptions), CRL also incorporates extended atomic patterns namely: *Else* and *ElseNext* patterns. These patterns giving the CRL expressions are conjunctively formed with *LeadTo* and *DirectlyFollowedBy* atomic patterns as:

$$\phi\ (\mathit{LeadsTo}|\mathit{DirectlyFollowedBy})\ \phi_1\ (\mathit{Else}|\mathit{ElseNext})\ \phi_2 \ldots (\mathit{Else}|\mathit{ElseNext})\ \phi_n \quad (5)$$

where ϕ is the rule condition, ϕ_1 is the primary action, and $\phi_2, \ldots, \phi_n$ are compensatory actions. Essentially, the compen-

sation pattern implements the *if-then-else* conditional structure of compensatory rules. Accordingly, *DirectlyFollowedBy* and *LeadsTo* define the ordering of the primary activity whether ϕ_1 directly occurs *immediately after* ϕ or it occurs sometimes in future. The LTL equivalence formula for compensatory patterns as follows:

$$\mathsf{G}(\phi \rightarrow \mathsf{F}|\mathsf{X}(\phi_1 \wedge_{1 \leq i < n-1} (\mathsf{F}|\mathsf{X}(\phi_i \ \mathit{NotSucceed}) \wedge (\phi_i \ \mathit{NotSucceed} \rightarrow \mathsf{F}|\mathsf{X}\phi_{i+1})))) \quad (6)$$

where ϕ gives the antecedent of the compensatory rule i.e., rule's conditions; ϕ_1 is the head of the rule representing the primary action that must be taken; $\phi_2, \ldots, \phi_n$ represents the compensatory actions that must be taken if the conditions of the rule are violated; and i is a natural number i.e., $n \in \mathbb{N}$; and ϕ_i *NotSucceed* represents the decision point that checks whether ϕ_i holds.

In addition to the generic rules patterns, CRL offers patterns for modelling non-monotonic requirements. This allows CRL to model *exceptions*. More specifically, exceptions provide conditions under which the primary requirement might not hold. Following [17] CRL has two patterns for exceptions: one for *strong exceptions* and one for *weak exceptions*: A strong exception on the primary rule mandates that whenever the strong exception holds, the primary rule *must* not hold. While a weak exception indicates that when the weak exception holds, the primary rule *might* or *might not* hold. The patterns for strong and weak exceptions as follows:

1) strong exception: $[[R]]$*Pattern*,
2) weak exception: $[R]$*Pattern*;

where $[[R]]$ and $[R]$ are the LTL formulas encoding the exception conditions and *Pattern* is the LTL formula corresponding to the primary requirement the exception applies to.

Given the potential recursive nature of exceptions (i.e., to be able to capture exceptions to exceptions), CRL recursively resolves the dependencies of the exception conditions, using the following translation to LTL

1) $[[R]]$*Pattern* is translated to $\phi \rightarrow \neg\psi$,
2) $[R]$*Pattern* is translated to $\phi \vee \psi$;

where ϕ is the LTL formula corresponding to R and ψ is the LTL counterpart of *Pattern*.

VI. A Privacy Act

In this section we introduce the scenario proposed in [9] to show limitations of temporal logic for the modelling of norms.

Suppose that a Privacy Act contains the following norms:[4]

Section 1. The collection of personal information is forbidden, unless acting on a court order authorising it.

Section 2. The destruction of illegally collected personal information before accessing it is a defence against the illegal collection of the personal information.

Section 3. The collection of medical information is forbidden, unless the entity collecting the medical information is permitted to collect personal information.

In addition the Act specifies what personal information and medical information are, and they turn out to be disjoint.

Suppose an entity, subject to the Act, collects some personal information without being permitted to do so; at the same time they collect medical information. The entity recognises that they illegally collected personal information (i.e., they collected the information without being authorised to do so by a Court Order) and decides to remediate the illegal collection by destroying the information before accessing it. Is the entity compliant with the Privacy Act above? Given that the personal information was destroyed the entity was excused from the violation of the first section (illegal collection of personal information). However, even if the entity was excused from the illegal collection of medical collection, they were never entitled (i.e., permitted) to collect personal information[5], consequently they were not permitted to collect medical information; thus the prohibition of collecting medical information was in force. Accordingly, the collection of medical information violates the norm forbidding such an activity.

VII. Privacy Act in CRL/LTL

In this section, first we are going to show how to represent the Privacy Act of Section VI using the CRL patterns presented in Section V, and then we combine the resulting representation with a simple business process model implementing the activity of collecting data. Based on the CRL representation we are going to assess if the process is compliant or not.

Following the analysis in [9] Section 1 establishes two conditions:

i. Typically the collection of personal information is forbidden; and
ii. The collection of personal information is permitted, if there is a court order authorizing the collection of personal information.

Section 2 can be paraphrased as follows:

iii. The destruction of personal information collected illegally before accessing it excuses the illegal collection.

Similarly to Section 2, Section 3 states two conditions:

iv. Typically the collection of medical information is forbidden; and
v. The collection of medical information is permitted provided that the collection of personal information is permitted.

Based on the above discussion, if we abstract from the actual contents of the norms, the structure of the act can be represented by the following set of norms (extended form):

E1. A is forbidden.
E2. A is permitted given C (alternatively: if C, then A is permitted).

[4]The Privacy Act presented here, though realistic, is a fictional one. However, (i) it is based on the novel Australian Privacy Principles (APP), Privacy (Enhancing Privacy Protection) Act 2012, and (ii) sections with the same logical structure as the clauses of this fictional act are present in the APP Act.

[5]If they were permitted to collect personal information, then the collection would have not been illegal, and they did not have to destroy it.

E3. The violation of A is compensated by B.
E4. D is forbidden.
E5. If A is permitted, so is D.

To compensate a violation we have to have a violation the compensation compensates. Moreover, to have a violation we have to have an obligation or prohibition, the violation violates. Accordingly, it makes sense to combine E1 and E3 in a single norm, obtaining thus the following set of norms (condensed form):

C1. A is forbidden; its violation is compensated by B.
C2. A is permitted given C (alternatively: if C, then A is permitted).
C3. D is forbidden.
C4. If A is permitted, so is D.

Based on the above analysis we can tackle the issue of how to represent the norms as CRL expressions based on the patterns of Section V. That something is forbidden means that it should not appear in the process, thus we can use the *isAbsent* pattern. For the compensation, the natural choice is to use the *Else*/*ElseNext* pattern. C2 and C4 set (weak) exceptions to the primary norms, C2 to the prohibition of the norm in C1, and C4 to the norm in C3. Accordingly, the first approximation is

CRL1. R_1: ($[R_2]A$ *isAbsent*) *Else* B,
CRL2. R_2: C,
CRL3. R_3: $[R_4]D$ *isAbsent*,
CRL4. R_4: A *isPermitted*.

First of all we like to point out that a prohibition corresponds to a maintenance obligation, and it is represented by as *isAbsent*. However, the first problem we have here is that the translation of *Else*/*ElseNext* pattern cannot be used for maintenance obligations. The translation given in (6) results in the following LTL formula

$$\mathsf{G}(\mathsf{F}|\mathsf{X}(\mathsf{G}\neg A \wedge \mathsf{F}|\mathsf{X}(A \wedge (A \rightarrow \mathsf{F}|\mathsf{X}B)))). \tag{7}$$

This formula is always false, given the conjunction of $\mathsf{G}\neg A$ and $\mathsf{F}|\mathsf{X}A$. The key reason why the pattern does not work for maintenance obligations and prohibitions is that the condition for a maintenance obligation for *not succeeding* is that the obligation has been violated, meaning that we have the opposite of the obligation (see Section VI, Definition 3, and notice that in a temporal logic setting $o \notin State(t,k)$ is equivalent to $\neg o \in State(t,k)$ or, in LTL parlance, $TS, t_k \vDash \neg o$). Consequently, as remarked in [9], a violation of a maintenance obligation is represented in LTL by the formula $\mathsf{G}\phi \wedge \neg\phi$. To obviate this problem we can use the solution advanced in [9], where the compensation of maintenance obligation is semantically defined as

$$\begin{aligned} TS, \sigma \vDash \phi \otimes \psi \text{ iff } \forall i \geq 0,\ TS, \sigma_i \vDash \phi;\ \text{or} \\ \exists j,k : 0 \leq j \leq k,\ TS, \sigma_j \vDash \neg\phi \text{ and } TS, \sigma_k \vDash \psi. \end{aligned} \tag{8}$$

Syntactically, this can be represented by the LTL formula

$$\mathsf{G}\phi \vee \mathsf{F}(\neg\phi \wedge \mathsf{F}|\mathsf{X}\psi). \tag{9}$$

The next issue we have to address is how to model permissions in CRL. Given that it is not possible to violate a permission, permissions seem not to play any role in compliance. CRL does not support specific patterns for the modelling of permissions. While it is true that permissions cannot be violated, and thus they are not needed for the semantics for compliance. To determine when a process is compliant one has to know what are the obligations in force for the various states traversed by the traces of the process. Accordingly, a domain expert who understands the regulatory requirement can populate the *Force* function based on her understanding of the legal framework the process is subject to. This means that in that approach one can dispense from a logical representation of the regulatory requirements. However, this approach becomes rapidly unattainable, given that, even for small to medium size business processes, the number of traces and states in the traces is large and so is the number of obligations and prohibitions ([11] reports on a real life case study with a medium size process, containing approximately 40 tasks, that would require to populate over 25,000 states, with over 100 obligations).

The discussion so far suggests that we need methods to (automatically) determine what obligations are in force given a set of regulatory requirements. Furthermore, the scenario given in Section VI demonstrates that permissions can play a role in compliance: they can be used as conditions that determine when other obligations or prohibitions are in force. For example, consider the compliant process where (1) the entity checks whether the collection of personal information is authorised under a court order. Then, if a court order exists (2) it proceeds to collect personal information, (3) it collects medical information. This process would be deemed as not compliant, since R_4 would resolve in D (collection of medical information) is absent but it occurs in the process.

As we have seen in Section VI in legal theory a permission is considered as the absence of obligation to the contrary. Thus, in deontic logic the deontic operator for permission (P) is assumed to be dual of the operator for obligation (O), i.e.:

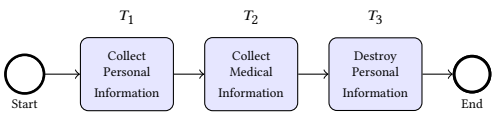
$$\mathsf{P}\phi \equiv \neg\mathsf{O}\neg\phi. \tag{10}$$

In the case at hand the obligation is a maintenance obligation, and, as we have argued, it corresponds to the *isAbsent* pattern, which is translated as $\mathsf{G}\neg A$; its dual is $\mathsf{F}A$. Thus, based on this analysis the translation from CRL to LTL gives the following two formulas:

LTL1. $\mathsf{G}(C \vee (\mathsf{G}\neg A \vee \mathsf{F}(A \wedge \mathsf{F}B)))$;
LTL2. $\mathsf{G}(\mathsf{F}A \vee \mathsf{G}\neg D)$.

CRL2 and CRL4 are incorporated in the translations of CRL1 and CRL3, i.e., LTL1 and LTL2 respectively.

Consider now the following process to collect information.



This process has a single trace, $\langle Start, T_1, T_2, T_3, End\rangle$. The transition system corresponding to this trace has the following transitions:[6]

$$(start, T_1), (T_1, T_2), (T_2, T_3), (T_3, end), (end, end) \qquad (11)$$

Suppose that for a particular instance of the process there is no court order authorising the collection of medical data, i.e., $\neg C$ holds for all the states reached by the execution of the process. Thus, the evaluation function associated to the trace is as follows:

- $v(start) = \{\neg A, \neg B, \neg C, \neg D\}$;
- $v(T_1) = \{A, \neg B, \neg C, \neg D\}$;
- $v(T_2) = \{A, \neg B, \neg C, D\}$;
- $v(T_3) = \{A, B, \neg C, D\}$;
- $v(end) = \{A, B, \neg C, D\}$.

It is easy to verify that the transition system corresponding to the trace of the process is a model of the formulas encoding the privacy act, LTL1 and LTL2. This means that the formulas are satisfied in all the states in it. For LTL1 we notice that the first disjunct, $\neg C$, is always false, but the second disjunct is always satisfied: every state in the transition system has a state following it where A holds, and there is a state following it where B holds. For LTL2, the first disjunct is true: for each state, there is a state following it where A holds, thus $\mathsf{F}A$ holds. Hence, the process is compliant with the LTL formulas encoding based on the CRL patterns modelling the privacy act. However, there is a state T_2 where both $\neg C$ and D hold. In Section VI we argued that a situation where $\neg C$ and D both hold is not compliant. Therefore, we have a paradox: the formalisation indicates that the scenario is compliant, the course of actions described by the transition system does not result in a contradiction, so no illegal action is performed (or better, the collection of personal information is illegal, but its compensation, the destruction of the personal information, makes full amend of it), but our legal intuition suggests that the collection of medical information in the circumstances of the scenario is illegal, and thus the process is not compliant.

The analysis in this section indicates that there is a mismatch between the outcome produced by the formal representation based on the CRL patterns modelling the legal requirements of the privacy act and the expected outcome according to the legal understanding of the underlying scenario. This means that the CRL patterns proposed by COMPAS cannot guarantee a correct assessment of whether a business process complies with a set of norms. The result is not restricted to CRL but it extends to other existing compliance frameworks based on temporal logic, since, essentially, they share the same occurrence and ordering patterns. The other frameworks do not have patterns to model exceptions and compensations, but [9] shows how to model such patterns in LTL.

[6]The transition (end, end) is mandated by the semantics of LTL that requires each state to have a successor; for the state corresponding to the termination of the process, the successor is itself.

VIII. Discussion

In this paper we have examined whether existing compliance frameworks based on temporal logic, in particular LTL, are able to represent norms in a conceptually sound way. The answer seems to be negative. The result is the consequence of the use of LTL as underlying formalism as highlighted in [9] and its inability to handle permissions.

We would like to point out that the discussion in the previous section just shows that *a* particular formalisation based on LTL is not suitable to represent the scenario, not that LTL *per se* is not able to represent the scenario. Indeed one could create all possible full paths in a transition system not breaching the norms, and then use the paths to synthesise the norms that regulate the transition system. However, we believe that such *ex post* analysis is useless. First humans have to perform the reasoning to determine which norms hold and when, and then which paths violate the norms. In a compliance perspective, i.e., in situations where one wants to determine if the specifications of a system comply with a set of norms, this approach requires to have an oracle able to discern the compliant executions (traces) form the non-compliant ones [14], and based on the oracle analysis to remove the traces resulting in violations. This means that LTL is not used for reasoning about the norms and the transition system, and verifying whether the transition system complies with the norms. But the strength of LTL is its ability to verify specifications against transition systems. However, in such a case, given that the specifications (i.e., the formalisation of the norms) are derived from the transition systems (which have been determined to be compliant by the oracle) the verification is always positive and totally uninformative. Furthermore, we believe that the formalisation we proposed, while naive, is extremely intuitive. The major objection is that permissions are modelled using F, and we hinted that F might be suitable to model achievement obligations, and using a particular type of obligation to model permissions is not appropriate and counter-intuitive outcomes are to be expected. We fully agree with this objection, but if we agree that a permission is the lack of an obligation to the contrary, then F is the natural choice for permissions for prohibitions (maintenance obligations). The other issue is that if we do not use F, then the issue is how to model permissions, and the alternative is that LTL does not support permissions. The act we presented clearly shows that there are acts where permissions must be represented and that permissions play an important role in determining which obligations are in force and when they are in force. Hence, any formalisation excluding permissions is doomed to be unable to represent the vast majority of real-life legal norms.

What about other patterns to model permissions, for example, introducing the justification of the reasons why A is permitted or the lack of the compensation. The first response to this objection is that, then, the formal representation would not correspond to the textual provision, broking the isomorphism principle for the formalisation of legal

knowledge [18], meaning that, again, an external oracle is needed to provide the proper formalisation of the conditions depending on the interpretation of the norm, the other norms, and the process. The second objection is that several of such patterns do not really solve the problem or introduce counter-intuitive results with other processes: for instance, using the formula $C \rightarrow \mathsf{F}A$ (saying that eventually A in case of C) as R is of no avail since in the case at hand $\neg C$ makes it true and we can ignore again the other disjunct.

It is worth noting that the problem depends on sets of LTL formulas with a particular structure (i.e., LTL1 and LTL2 presented in this paper or the set of formulas discussed in [9]), and the natural reading of such expressions, in particular when the deontic interpretation is imposed to the temporal logic operators and their combination based on the definitions given in Section III.

Next, a natural question is whether branching time logics with path quantifiers such as CTL and CTL* are more apt for this task.[7] In such logics permissions could be formalised by EF. While modelling permissions using path quantifiers seems a better option and provides more flexibility for modelling norms, it does not however solve the problem with the scenario we proposed, given that the problem requires just a single (non) branching trace to arise; thus path quantifiers are essentially irrelevant.

This paper provides a (realistic) set of norms that shows that current approaches to business process compliance based on temporal logic patterns are not suitable for the task at hand (i.e., to determine whether a process complies with the norms), since the result they provide is not aligned with the expected outcome based on the legal interpretation of the scenario. The reason being that they fail to represent and reason with the norms in a conceptually sound way. Accordingly, they cannot be used to check compliance of real business processes with real-life norms.

The final issue we want to discuss is whether there are compliance frameworks that provide conceptually sound approaches. [11] points out problems of other compliance frameworks based on event-calculus, e.g., [20], or first-order logic, for example [21]. Thus, such approaches would not be suitable to model compliance with real-life norms. We want to remark that the paradox, discussed in Section VII, is not restricted to LTL or other temporal logics. It can be easily replicated in Standard Deontic Logic (and it is well known that Standard Deontic Logic is plagued with many other contrary-to-duty paradoxes). A root-cause analysis of the paradox is that a violation of a compensable obligation results in a sub-ideal state. Hence, there is a state with a violation that is still deemed legal. This means, that there is a (somehow) legal state, and if permission is evaluated as being in at least one legal state, then the violation has to be evaluated as (somehow) permitted. Part of the problem is that in such somehow legal states there might be other true legitimate permissions which are not the violation of compensable obligations. Accordingly, we conjecture, that logics using truth of a formula in at least one (somehow) legal state to determine whether something is permitted have counterparts of the paradox we presented. Governatori and Rotolo [22] proposed a non-boolean (sub-structural) operator $\otimes$ to model compensatory obligations. [23] proposes a novel possible world semantics for the operator, and shows that it is immune from the paradox. The semantics distinguish norms from obligations and permissions: a permission can be derived if there is an explicit norms that permits it. The operator has been incorporated in defeasible logic in [24], and the resulting logic, FCL, has been used as the foundation of the *compliance-by-design* methodology proposed by Governatori and Sadiq [25], [26] and compliance checking algorithm of [27]. The methodology has been implemented in the Regorous compliance checker[8], that has been successfully tested on industry scale use cases [28].

[7] For a discussion of the pros and cons of linear and branching time logics, we refer to [19], though such discussion is not on their capability of modelling norms.

Acknowledgments

We thank Nick van Beest and the anonymous referees for their valuable comments on a previous draft of the paper.

NICTA is funded by the Australian Government and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] A. Pnueli, "The temporal logic of programs," in *SFCS '77: Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, 1977, pp. 46–57.

[2] A. Elgammal, O. Türetken, W.-J. van den Heuvel, and M. Papazoglou, "On the formal specification of regulatory compliance: A comparative analysis," in *Service-Oriented Computing*, ser. LNCS, vol. 6568, Springer, 2011, pp. 27–38. DOI: 10.1007/978-3-642-19394-1_4.

[3] ——, "Formalizing and applying compliance patterns for business process compliance," *Software & Systems Modelling*, 2014. DOI: 10.1007/s10270-014-0395-3.

[4] A. Awad, M. Weidlich, and M. Weske, "Visually specifying compliance rules and explaining their violations for business processes," *Journal of Visual Languages and Computing*, vol. 22, no. 1, pp. 30–55, 2011.

[5] A. Awad, R. P. Goré, Z. Hou, J. Thomson, and M. Weidlich, "An iterative approach to synthesize business process templates from compliance rules," *Information Systems*, vol. 37, no. 8, pp. 714–736, 2012. DOI: 10.1016/j.is.2012.05.001.

[6] C. Giblin, A. Y. Liu, S. Müller, B. Pfitzmann, and X. Zhou, "Regulations expressed as logical models (REALM)," in *Proceeding of the 2005 conference on Legal Knowledge and Information Systems: JURIX 2005: The Eighteenth Annual Conference*, IOS Press, 2005.

[7] M. Pesic, H. Schonenberg, and W. M. P. van der Aalst, "DECLARE: Full support for loosely-structured processes," in *Enterprise Distributed Object Computing Conference, 2007*, IEEE Computer Society, 2007, pp. 287–300. DOI: 10.1109/EDOC.2007.14.

[8] R. H. Thomason, "Deontic logic founded on tense logic," in *New Studies on Deontic Logic*, R. Hilpinen, Ed., Kluwer, 1981, pp. 165–176.

[9] G. Governatori, "Thou shalt is not you will," in *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Law*, K. Atkinson, Ed., New York: ACM, 2015, pp. 63–68. DOI: 10.1145/2746090.2746105.

[10] ——, "Business process compliance: An abstract normative framework," *IT – Information Technology*, vol. 55, no. 6, pp. 231–238, 2013. DOI: 10.1515/itit.2013.2003.

[11] M. Hashmi, G. Governatori, and M. T. Wynn, "Normative requirements for regulatory compliance: An abstract formal framework," *Information Systems Frontiers*, 2015. DOI: 10.1007/s10796-015-9558-1.

[8] https://www.regorous.com/.

[12] W. M. P. van der Aalst, "Workflow verification: Finding control-flow errors using Petri-net-based techniques," in *Business Process Management: Models, Techniques, and Empirical Studies*, W. M. P. van der Aalst, J. Desel, and A. Oberweis, Eds., ser. LNCS, vol. 1803, Springer, 2000, pp. 161–183.

[13] T. Athan, G. Governatori, A. Paschke, M. Palmirani, and A. Wyner, "LegalRuleML: Design principles and foundations," in *Reasoning Web. Web Logic Rules*, ser. LNCS 9203, W. Faber and A. Paschke, Eds., Springer, 2015, pp. 151–188. DOI: 10.1007/978-3-319-21768-0_6.

[14] G. Governatori and S. Sadiq, "The journey to business process compliance," in *Handbook of Research on BPM*, J. Cardoso and W. van der Aalst, Eds., IGI Global, 2009, ch. 20, pp. 426–454.

[15] G. Sartor, *Legal Reasoning: A Cognitive Approach to the Law*. Springer, 2005.

[16] M. Dwyer, G. Avrunin, and J. Corbett, "Patterns in property specifications for finite-state verification," in *Software Engineering, 1999. Proceedings of the 1999 International Conference on*, 1999, pp. 411–420.

[17] C. Baral and J. Zhao, "Non-monotonic temporal logics for goal specification," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers Inc, 2007, pp. 236–242.

[18] T. Bench-Capon and F. P. Coenen, "Isomorphism and legal knowledge based systems," *Artificial Intelligence and Law*, vol. 1, no. 1, pp. 65–86, 1992.

[19] M. Y. Vardi, "Branching vs. linear time: Final showdown," in *7th International Conference Tools and Algorithms for the Construction and Analysis of Systems, (TACAS 2001)*, T. Margaria and W. Yi, Eds., ser. LNCS, Springer, 2001, pp. 1–22. DOI: 10.1007/3-540-45319-9_1.

[20] S. Goedertier and J. Vanthienen, "Designing compliant business processes with obligations and permissions," in *BPM Workshops*, ser. LNCS, vol. 4103, Springer, 2006, pp. 5–14.

[21] L. T. Ly, S. Rinderle-Ma, K. Göser, and P. Dadam, "On enabling integrated process compliance with semantic constraints in process management systems," *Information Systems Frontiers*, vol. 14, no. 2, pp. 195–219, 2012.

[22] G. Governatori and A. Rotolo, "Logic of violations: A Gentzen system for reasoning with contrary-to-duty obligations," *Australasian Journal of Logic*, vol. 4, pp. 193–215, 2006.

[23] E. Calardo, G. Governatori, and A. Rotolo, "A sequence semantics for deontic logic," NICTA, Tech. Rep. 8580, 2015.

[24] G. Governatori, "Representing business contracts in RuleML," *International Journal of Cooperative Information Systems*, vol. 14, no. 2-3, pp. 181–216, 2005.

[25] S. Sadiq, G. Governatori, and K. Naimiri, "Modelling of control objectives for business process compliance," in *BPM 2007*, G. Alonso, P. Dadam, and M. Rosemann, Eds., ser. LNCS, Berlin: Springer, 2007, pp. 149–164. DOI: 10.1007/978-3-540-75183-0_12.

[26] S. Sadiq and G. Governatori, "Managing regulatory compliance in business processes," in *Handbook of Business Process Management 2nd edition*, ser. International Handbooks on Information Systems, J. vom Brocke and M. Rosemann, Eds., vol. 2, Berlin-Heidelberg: Springer, 2015, pp. 265–288. DOI: 10.1007/978-3-642-45103-4_11.

[27] G. Governatori and A. Rotolo, "A conceptually rich model of business process compliance," in *7th Asia-Pacific Conference on Conceptual Modelling*, S. Link and A. Ghose, Eds., ser. CRPIT, vol. 110, ACS, 2010, pp. 3–12.

[28] G. Governatori and S. Shek, "Regorous: A business process compliance checker," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law*, E. Francesconi and B. Verheij, Eds., New York: ACM, 2013, pp. 245–246. DOI: 10.1145/2514601.2514638.