2015 IEEE 19th International Enterprise Distributed Object Computing Workshop

The Regorous Approach to Process Compliance

Guido Governatori
NICTA, Brisbane, Australia
Email: guido.governatori@nicta.com.au

***Abstract*—We propose an ITC (Information and Communication Technology) approach to support regulatory compliance for business processes, and we report on the development and evaluation of a business process compliance checker called Regorous, based on the compliance-by-design methodology proposed by Governatori and Sadiq [1].**

I. Introduction

Regulatory compliance is the set of activities an enterprise does to ensure that its core business does not violate relevant regulations, in the jurisdictions in which the business is situated, governing the (industry) sectors where the enterprise operates.

The activities an organisation does to achieve its business objectives can be understood as business processes, and consequently they can be represented by business process models. On the other hand a normative document (e.g., a code, a bill, an act) can be understood as a set of clauses, and these clauses can be represented in an appropriate formal language. Based on this [2] proposed that *business process compliance* is a relationship between the formal representation of a process model and the formal representation of the relevant regulations.

Obviously, any approach to automate the checking whether a business process complies with the regulation governing has to ensure that it is able to properly model business processes as well as norms. In the past decades many approaches to automatise business process compliance have been proposed. [3], [4] survey the state of the art and list expected functionalities and other desiderata for compliance framework. Surprisingly, the current research fails to address the most fundamental questions: are current compliance management frameworks (CMFs) able to model norms in a conceptually sound way? [5] provides a comprehensive classifications of the class of the normative concepts (e.g., obligations, prohibitions, permissions,…) required for business process compliance, and introduces their semantics in terms of business processes. Furthermore, for each requirement it shows examples from real life acts where the requirement appears. [6] investigated which CMFs provide direct natural counterparts of the normative concepts identified in [5]. The answer is that, apart some notable exceptions, the coverage is very limited. This does not mean that a CMF with a limited coverage is not able to capture the semantics of the normative concepts, but that end users have to rely on deep understanding of the formalisms the frameworks are based on instead of the templates they provide. This shift focus from the frameworks to the underlying formalisms. Temporal logics (e.g., LTL and CTL) and Event Calculus have been used in several CMFs. However, [7] identifies several shortcomings in using Event Calculus for the modelling norms. Recently, [8] shows that when norms are formalised in Linear Temporal Logi (or, more in general, in logics in the same family), the evaluation whether a process is compliant produces results that are not compatible with the intuitive and most natural legal interpretation. Furthermore, [8] argues that, while such logics can properly model norms such formalisations would be completely useless from a process compliance point of view insofar they would require an external oracle to identify the compliant executions of the process, and build the formalisation from the traces corresponding to the traces deemed legal by the oracle. Accordingly, this means that there is no need of the formalisation to determine if the process is compliant or not, since this is done by the oracle. In any case [9] shows that current CMFs based on temporal logic are affected by the scenario given in [8]. Consequently, the solutions provided by such CMFs cannot be used to check compliance of real life business processes with real life norms.

The contribution of this paper is to show that Regorous, a compliance framework based on the compliance-by-design methodology proposed by Governatori and Sadiq [1], [10], is immune from the problem of not being able to correctly assess compliance of business processes in scenarios involving permissions [8], [9] affecting the existing compliance frameworks based on temporal logic.

The paper is organised as follows: Sections II, III and IV outline the formal foundations for modelling business process compliance. Section V provides a concise presentation of Regorous and its architecture, and we quickly report on the results of an empirical evaluation of the framework with an industry scale pilot project (Section V-A). In Section VI we introduce the scenario proposed in [8] and used in [9] to show limitations of compliance frameworks based on temporal logic. We analyse how the scenario is modelled in Regorous, and we show that Regorous produces the correct assessment of compliance. This indicates that Regorous does not suffer from the problem affecting other compliance frameworks. Finally, we are going to argue that while Regorous is specifically created for design-time compliance, the approach is equally valid for run-time compliance and auditing based on process logs (Section VII).

II. Business Process Modelling

In this section we provide the very basics of business process modelling, for an extensive presentation see [11].

978-1-4673-9331-7/15 $31.00 © 2015 IEEE
DOI 10.1109/EDOCW.2015.28

A business process model is a self-contained, temporal and logical order in which a set of activities are expected to be 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). Many different formalisms (Petri-Nets, Process algebras, …) and notations (BPMN, YAWL, EPC, …) have been proposed to represent business process models. Besides the difference in notation, purposes, and expressive power, business process languages typically contain the following minimal set of elements: *tasks*, *connectors* (control flow gateways) and *events*. A task corresponds to a (complex) business activity, and connectors (e.g., sequence, and-join, and-split, (x)or-join, (x)or-split) define the relationships among tasks to be executed; for the events we restrict ourselves to the start and end event. The combination of tasks and connectors defines the possible ways in which a process can be executed. Where a possible execution, called *process trace* or simply *trace*, is a sequence of tasks and events respecting the order given by the connectors.

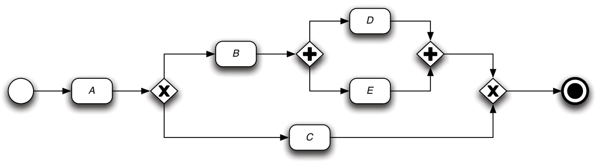

Figure 1. Example of a business process model in standard BPMN notation

Consider the process in Figure 1, in standard BPMN notation, where we have a task A followed by an xor split. In the xor split in one of the branches we have task B followed by the and-split of a branch with task D, and a brach consisting of only task E. The second branch of the xor-split has only one task: C. The traces corresponding to the process are

$$\langle start, A, C, end\rangle,$$
$$\langle start, A, B, D, E, end\rangle,$$
$$\langle start, A, B, E, D, end\rangle.$$

Given a process P we will use $\mathcal{T}_P = \{t_1, t_2, \dots\}$ to denote the set of traces of P.

Compliance is not only about the tasks that an organisation has to perform to achieve its business goals, but it is concerned also on their effects (i.e., how the activities in the tasks change the environment in which they operate), and the artefacts produced by the tasks (for example, the data resulting from executing a task or modified by the task) [12]. To capture this aspect [13] proposed to enrich process models with semantic annotations. Each task in a process model can have attached to it a set of semantic annotations. An annotation is just a set of formulas giving a (partial) description of the environment in which a process operates. Then, it is possible to associate to each task in a trace 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 results 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, we introduce the function

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

where $\mathbb{N}$ is the set of natural numbers and $\mathcal{L}$ is the set of formulas of the language used to model the annotations. 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 we can say that p, q and r are the effects of task D in the trace, and thus p, q and r hold after the execution of D in 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 events), and the corresponding sequence of states.

III. Normative Requirements

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. A comprehensive list of normative effects is provided in [14]. In a compliance perspective, the normative effects of importance are the deontic effects (also called normative positions). The basic deontic effects are: *obligation*, *prohibition* and *permission*.[1]

Let us start by consider 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 prohibition of it or the obligation to the contrary do not hold.

Obligations and prohibitions are constraints that limit the space of action 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

[1] There are other deontic effects, but these can be derived from the basic ones, see [15].

[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.

can be compensated for, and processes with compensated violations are still compliant [1], [16]; for example contracts typically contain compensatory clauses specifying penalties and other sanctions triggered by breaches of other contract clauses [17]. Not all violations are compensable, and uncompensated violations means that a process is not compliant. Permissions cannot be violated, thus permissions do not play a direct role in compliance; they can be used to determine that there are no obligations or prohibitions to the contrary, or to derive other deontic effects. 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]. In this paper we will subscribe to this position, given that our focus here is not on how to determine what is prescribed by a set of norms and how to derive it. 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 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. The question then is how long does an obligation hold for, and based on this there are different conditions to fulfil the obligation. We take a systematic approach to this issue. A norm can specify that an obligation is in force for a particular time point or, more often, a norm indicates when an obligation enters in force. An obligation remains in force until terminated or removed. Accordingly, in the first case we will speak of *punctual obligations* and in the second case of *persistent obligations*.

For persistent obligations we can ask 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 final aspect we want to touch upon in this section is the termination of obligations. Norms can specify the interval in which an obligation is in force. Previously, we discussed that what differentiates obligations and other constraints is that obligations can be violated. What are the effects of a violation on the obligation the violation violates? More precisely, does a violation terminate the violated obligation? Meaning, do we still have to comply with a violated obligation? If we do –the obligation persists after being violated– we speak of a *perdurant obligation*, if it does not, then we have a *non-perdurant obligation*.

It is worth noticing that the classification discussed above is exhaustive. It has been obtained in a systematic and comprehensive way when one considers the aspect of the validity of obligations –or prohibitions– (i.e., whether they persist after they enter in force or they are valid only for a specific time unit), and the effects of violations on them, namely: whether a violation can be compensated for, and whether an obligation persists after being violated.

The semantics for the various types of obligations is based on the function *Force* with the following signature:

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

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 fulfil to comply with a given normative framework; for instance $o \in t, 3$ specifies that o is obligatory for the third tasks in trace t.

The various types of obligations can be defined using combinations of the *Force* and *State* functions to specify when an obligation is in force and what constitutes a violation of the obligation. For example, maintenance obligations are defined as follows.

Definition 1 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

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

Definitions and real examples for all normative concepts outlined in this section are given in [5], [6].

IV. Modelling Compliance

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 2 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) obligations, 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 3 A trace t complies with a normative system $\mathcal{N} = \{n_1, n_2, \dots\}$ iff all norms n_i in $\mathcal{N}$ have not been violated.

In Section III we provided various types of norms. 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 3. Weak compliance is defined as follows:

Definition 4 A trace t weakly complies with a normative system $\mathcal{N}$ iff every violated norm has been compensated for.

V. Regorous Architecture

In this section we first introduce the architecture of Regorous Process Designer[3] (from now on simply Regorous), a business process compliance checker based on the business process compliance methodology proposed by Governatori and Sadiq [1], [10].

As we have already discussed to check whether a business process is compliant with a relevant regulation, we need an annotated business process model and the formal representation of the regulation. The annotations are attached to the tasks of the process, and it can be used to record the data, resources and other information related to the single tasks in a process.

For the formal representation of the regulation we use FCL [17], [18]. FCL is a simple, efficient, flexible rule based logic. FCL has been obtained from the combination of defeasible logic (for the efficient and natural treatment of exceptions, which are a common feature in normative reasoning) [19] and a deontic logic of violations [20]. In FCL a norm is represented by a rule

$$a_1, \dots, a_n \Rightarrow c$$

Where $a_1, \dots, a_n$ are the conditions of applicability of the norm/rule and c is the *normative effect* of the norm/rule. FCL distinguishes two normative effects: the first is that of introducing a definition for a new term. For example the rule

$$\mathit{customer}(x), \mathit{spending}(x) > 1000 \Rightarrow \mathit{premium\_customer}(x)$$

specifies that, typically, a premium customer is a customer who has spent over 1000 dollars. The second normative effect is that of triggering obligations and other deontic notions. FCL supports all deontic notions presented in Section III, in addition it has mechanisms to terminate and remove obligations (see [18] for full details). For obligations and permission we use the following notation:

- $[\mathrm{P}]p$: p is permitted;
- $[\mathrm{OM}]p$: there is a maintenance obligation for p;
- $[\mathrm{OAPP}]p$: there is an achievement preemptive and perdurant obligation for p;
- $[\mathrm{OAPNP}]p$: there is an achievement preemptive and non-perdurant obligation for p;
- $[\mathrm{OANPP}]p$: there is an achievement non preemptive and perdurant obligation for p;
- $[\mathrm{OANPNP}]p$: there is an achievement non preemptive and non-perdurant obligation for p.

Compensations are implemented based on the notion of 'reparation chain' [20]. A reparation chair is an expression

$$\mathrm{O}_1 c_1 \otimes \mathrm{O}_2 c \otimes \cdots \otimes \mathrm{O}_n c_n,$$

where each O_i is an obligation, and each c_i is the content of the obligation (modelled by a literal). The meaning of a reparation chain is that we have that c_1 is obligatory, but if the obligation of c_1 is violated, i.e., we have $\neg c_1$, then the violation is compensated by c_2 (which is then obligatory). But if even $\mathrm{O}_2 c_2$ is violated, then this violation is compensated by c_3 which, after the violation of c_2, becomes obligatory, and so on.

It is worth noticing that FCL allows deontic expressions (but not reparation chains) to appear in the body of rules, thus we can have rules like:

$$\mathit{restaurant}, [\mathrm{P}]\mathit{sell\_alcohol} \Rightarrow [\mathrm{OM}]\mathit{show\_license} \otimes \\ [\mathrm{OAPNP}]\mathit{pay\_fine}.$$

The rule above means that if a restaurant has a license to sell alcohol (i.e, it is permitted to sell it, $[\mathrm{P}]\mathit{sell\_alcohol}$), then it has a maintenance obligation to expose the license ($[\mathrm{OM}]\mathit{show\_license}$), if it does not then it has to pay the fine ($[\mathrm{OAPNP}]\mathit{pay\_fine}$). The obligation to pay the fine is non-pre-emptive (meaning that it cannot be paid before the violation). FCL is equipped with a binary relations over rules, called superiority relation, that allows us to handle rules with conflicting conclusions: for example a rule r setting a general prohibition and a second rule s that derogates the prohibition permitting the conclusions. This type of situation is common in legal reasoning and can be modelled by saying that s is "stronger" than r, in symbols $s > r$. If both rules apply we will say that s defeats r. For full a description of FCL and its features, see [17], [18].

The reasoning to determine what obligations, prohibitions and permissions are derivable from a set of facts and a set or rules is as follows.

An obligation $[\mathrm{O}]p$ (where $[\mathrm{O}]$, $[\mathrm{O}_x]$ and $[\mathrm{D}_y]$, in the description below, are placeholders for the obligations described above) is derivable if:

1) $[\mathrm{O}]p$ is given as one of the facts, or
2) there is a rule

$$r\colon a_1, \dots a_n \Rightarrow [\mathrm{O}_1]p_1 \otimes [\mathrm{O}_m]p_m \otimes [\mathrm{O}]p \dots$$

such that

a) for all $1 \leq i \leq n$, a_i is provable, and
b) for all $1 \leq j \leq m$, $[\mathrm{O}_j]p_j$ and $\neg p_j$ are provable, and
c) for all rules

$$s\colon b_1, \dots, b_k \Rightarrow [\mathrm{D}_1]q_1 \otimes [\mathrm{D}_l]q_l \otimes [\mathrm{D}]p'$$

such that p' is the negation of p, either

i) exists $1 \leq i \leq k$ such that b_i is not provable, or
ii) exists $1 \leq j \leq l$ such that either $[\mathrm{D}_j]q_j$ or $\neg q_j$ is not provable, or

[3] Regorous is available under an evaluation license from http://www.regorous.com.

iii) r defeats s.

The idea is that there must be a rule that fires: so all the elements in the antecedents are provable (a), and in case the conclusion is an obligation for a reparation, all the obligations before it have to be violated. Thus, the violated obligation were in force (thus the obligations were provable) and we have evidence that it was violated (thus the negation of the content of each violated obligation is provable) (b). In addition, we have to ensure that there are no rules for the opposite that fire (c), and if they do, these rules are weaker than the rule for the obligation we want to conclude.

For permission, we have the same conditions, but where we use $[\mathrm{P}]p$ instead of $[\mathrm{O}]p$; also, we conclude $[\mathrm{P}]p$ if we can conclude $[\mathrm{O}]p$. For the full presentation of the logic we refer to [18], [21].

Finally, FCL is agnostic about the nature of the literals it uses. They can represent tasks (activities executed in a process), or propositions representing state variables and the happening of events.

Compliance is not just about the tasks to be executed in a process but also on what the tasks do, the way they change the data and the state of artefacts related to the process, and the resources linked to the process. Accordingly, process models must be enriched with such information. [13] proposes to enrich process models with semantic annotations. Each task in a process model can have attached to it a set of semantic annotations. In our approach the semantic annotations are literals in the language of FCL, representing the effects of the tasks. The approach can be used to model business process data compliance [12].

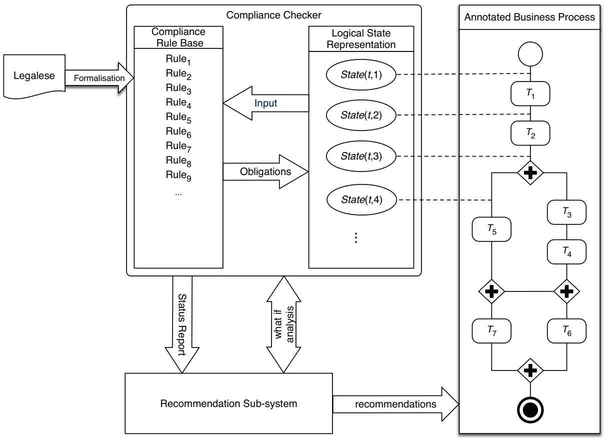

Figure 2. Architecture of Regorous

Figure 2 depicts the architecture of Regorous. Given an annotated process and the formalisation of the relevant regulation, we can use the algorithm initially proposed in [22] and then extended to cover the full FCL language in [18] to determine whether the annotated process model is compliant. The procedure runs as follows:

- Generate an execution trace of the process.
- Traverse the trace:
  - for each task in the trace, cumulate the effects of the task using an update semantics (i.e., if an effect in the current task conflicts with previous annotation, update using the effects of the current tasks).
  - use the set of cumulated effects to determine which obligations enter into force at the current tasks. This is done by a call to an FCL reasoner.
  - add the obligations obtained from the previous step to the set of obligations carried over from the previous task.
  - determine which obligations have been fulfilled, violated, or are pending; and if there are violated obligation check whether they have been compensated.
- repeat for all traces.

Thus for the n-th element of a trace t we use $\mathit{State}(t,n)$ as the set of facts in the computation to determine which rules fire, and consequently which obligations are in $\mathit{Force}(t,n+1)$. In addition, $\mathit{Force}(t,n)$ contains the obligations that are in $\mathit{Force}(t,n)$ but not terminated at n (an obligation can be terminated for the following three reasons: we reach its deadline, the obligation has been fulfilled, the obligation has been violated and it is not perdurant). To assess which obligations have been complied with or violated in n we compare the elements of $\mathit{Force}(t,n)$ and $\mathit{State}(t,n)$.

A process is (fully) compliant if and only if all traces are compliant (all obligations have been fulfilled or if violated they have been compensated). A process is partially compliant if there is at least one trace that is compliant.

Notice that the Regorous's strategy to examine all traces (with the proviso that all loops are unfolded once) is optimal, in the sense that [23] proved that the problem of determining whether a process is weakly compliant is NP-complete and CoNP-complete for the case of full compliance.

A. Implementation and Evaluation

Regorous Process Designer has been implemented on top of Eclipse. For the representation of process models, it uses the Eclipse Activiti BPMN 2.0 plugin, extended with features to allow users to add semantic annotations to the tasks in the process model. Regorous is process model agnostic, this means that while the current implementation is based on BPMN all Regorous needs is to have a description of the process and the annotations for each task. A module of Regorous take the description of the process and generates the execution traces corresponding to the process. After the traces are generated, it implements the algorithm outlined above, where it uses the SPINdle rule engine [24] for the evaluation of the FCL rules. In case a process is not compliant (or if it is only weakly compliant) Regorous reports the traces, tasks, rules and obligations involved in the non compliance issues.

Regorous was successfully tested against the 2012 Australian Telecommunications Customers Protection Code (C628-2012), in an industry scale pilot project in collaboration with an industry partner operating in the sector. See [10], [25] for the results of the evaluation.

VI. A Privacy Scenario

In this section we recall the scenario and related analysis presented in [8] to show that LTL cannot be used to model norms, and used in [9] to demonstrate that current CMFs based on temporal logic produce an incorrect compliance outcome.

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, 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.

Let us examine the structure of the act:

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 authorising 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 content of the norms, the structure of the act can be represented by the following set of norms (extended form):

E1. A ("collection of medical information") is forbidden.

E2. A is permitted given C ("acting under a court order"); alternatively: if C, then A is permitted.

E3. The violation of A is compensated by B ("destruction of collected medical information").

E4. D ("collection of personal information") 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.

Let us consider what are the situations compliant with the above set of norms. Clearly, if C does not hold, then we have that the prohibition of A and prohibition of D are in force. Therefore, a situation where $\neg A$, $\neg C$, and $\neg D$ hold is fully compliant (irrespective whether B holds or not). If C holds, then the permission of A derogates the prohibition of A, thus situations with either A holds or $\neg A$ holds are compliant with the first two norms; in addition, the permission of A allows us to derogate the prohibition of D. Accordingly, situations with either D or $\neg D$ comply with the third norm. Let us go back to scenarios where C does not hold, and let us suppose that we have A. This means that the prohibition of A has been violated; nevertheless the set of norms allows us to recover from such a violation by B. However, as we just remarked above to have a violation we have to have either an obligation or a prohibition that has been violated: in this case the prohibition of A. Given that the prohibition of A and the permission of A are mutually incompatible, we must have, to maintain a consistent situation, that A is not permitted. But if A was not permitted D is not permitted either; actually, according to the third norm, D is forbidden. To sum up, a scenario where $\neg C$, A, B and $\neg D$ hold is still compliant (even if to a lesser degree given the compensated violation of the prohibition of A). In any case, no situation where both $\neg C$ and D hold is compliant.

The Act can be modelled in FCL by the following theory:

FCL1 $r_1\colon\ \Rightarrow [\mathsf{OM}]\neg A \otimes [\mathsf{OANPP}]B;$

FCL2 $r_2\colon C \Rightarrow [\mathsf{P}]A;$

FCL3 $r_3\colon\ \Rightarrow [\mathsf{OM}]\neg D;$

FCL4 $r_4\colon [\mathsf{P}]A \Rightarrow [\mathsf{P}]D;$

FCL5 $r_2 > r_1,\ r_4 > r_3.$

[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 Amendment (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.

Table I
Evolution of the values of the compliance functions for the privacy scenario

Task/Event	*Chains*	*Force*	*Violated*	*Compensated*
start				
T_1	[OM]$\neg A \otimes$ [OANPP]B, [OM]$\neg D$	[OM]$\neg A$, [OM]D	[OM]$\neg A$	
T_2	[OANPP]B, [OM]D	[OM]$\neg A$, [OM]$\neg D$, [OANPP]B	[OM]$\neg A$, [OM]$\neg D$	
T_3	[OANPP]B	[OM]$\neg A$, [OM]$\neg D$, [OANPP]B	[OM]$\neg A$, [OM]$\neg D$	[OM]$\neg A$
end			[OM]$\neg A$, [OM]$\neg D$	[OM]$\neg A$

FCL1–FCL4 are a direct translation of the norms as presented in their condensed form. The two instances of the superiority relation in FCL5 indicate that the permissive norms C2 and C4 are exceptions to the norms in C1 and C3, respectively.

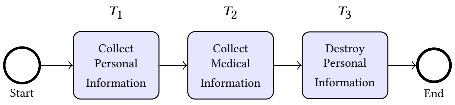
T_1
T_2
T_3
Collect Personal Information
Collect Medical Information
Destroy Personal Information
Start
End

Figure 3. Simple Information Collection Process for the Privacy Scenario

Consider the process in Figure 3. Clearly, the process consists of a single trace

$$t = \langle start, T_1, T_2, T_4, end \rangle,$$

and suppose that there is no court order authorising the collection of personal information. Accordingly, the *State* function is instantiated as follows:[6]

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

For each state Regorous maintains four sets related to the obligations and permissions: *Chains*, *Force*, *Violated* and *Compensated*.

Chains contains the conclusions (or fragments of them) of the undefeated rules that fire in the current state, or that where in *Chains* in the previous state and have not been terminated.

Force contains the obligations and prohibitions in force at the current state. These are the conclusions obtained by using *State* computed at the previous step as the set of facts used as input for the FCL rules (see the outline of the FCL mechanism in the previous section). In addition it contains the obligations and prohibitions from the previous iteration of the computation that were not terminated.

Violated cumulates the obligations that have been violated up to the current state in the process.

Compensated maintains the set of obligations which have been compensated.

Table I shows how the four sets above are populated for the process at hand.

Given that, $\neg C$ holds from *start*, rule r_1 fires, and from it we can derive the chain [OM]$\neg A \otimes$ [OANPP]B, thus the obligation (prohibition) [OM]$\neg A$ is in force from the first task in the process, namely T_1. For the same reason we are not able to conclude that [P]A holds (rule r_2 would require C to fire), thus we cannot use rule r_4, so rule r_3 fires and it is not defeated, thus we conclude [OM]$\neg D$. In T_1 we have $\neg A$, thus we have the violation of the prohibition to collect personal information. This means that from the next step the compensatory obligation [OANPP]B will be in force. In T_2, we have D, which implies that the prohibition of collecting medical information (i.e., [OM]$\neg D$) has been violated. In T_3 we introduce B. This means that the corresponding obligation [OANPP]B is fulfilled. In addition this compensates the violation of the obligation of [OM]$\neg A$. In the last step (*end*) we have to check whether there are pending achievement obligations (which is not the case) and whether all violated obligations have been compensated for. The violation of the prohibition of A has been compensated, but the violation of the prohibition of D has not. This means that, as far as A is concern, the process is weakly compliant, and the process is not compliant when we consider D. The result produced by FCL is fully aligned with the intuitive legal analysis of the scenario.

VII. Compliance at Design Time, Run Time and Auditing

The methodology and tool presented in the previous sections are primarily meant to help in the design of compliant business processes according the principle of compliance-by-design. While the tool is implemented in a computer system the proposed approach does not require the processes to be implement and executed by a workflow engine. Obviously, an enterprise obtain the major benefit when the tasks in a process are fully automated and the coordination of the order of execution of the task is under the control of a process-aware information system.Then, assuming a faithful implementation of the processes, all instances of the process are guaranteed to be compliant removing, potentially, the need of run-time monitoring and post-execution auditing.

At the other extreme of the spectrum we have the case where processes are not implemented by workflow engines. The proposed approach is still useful in so far as it can be used to establish the blue-prints of compliant processes. Clearly, if the tasks are executed by human operators (and the operators have flexibility about what operations are execute, and when to execute them), the tool cannot be used to support run-time

[6] To improve the readability we use the name of the tasks and events instead of the ordinal number.

monitoring and auditing, and other well establish methods have to used.

The last situation to consider is when there are no well defined process models, but the business activities (i.e., processes) are still supported by ICT technology in the form of recording of business events and message passing, and writing them in a log. In this scenario, the approach we proposed approach can be still applied. As we have outlined in Section IV Regorous simulates all the possible (finite) executions of a process, where an execution or trace is the sequence of tasks to be executed. In this case we can use a business event as a tasks. Here, instead of annotation the tasks in a process, we do the same on the business events and messages to be recored in the log, and extract the data using the techniques presented in [12]. At run-time, after each business events Regorous can compute what are the obligations, prohibitions in force after the business event, and evaluate whether they have been fulfilled or violated and report the resulting state. For auditing, Regorous can examine the log, and for each instance, replay it to determine, using the same algorithms for compliance, whether the instance was properly executed, and if it was compliant.

VIII. Conclusions

We reported on the development of a tool, Regorous Process Designer, for checking the compliance of business processes with relevant regulations, based on the compliance-by-design methodology proposed by Governatori and Sadiq [1]. Regorous is based on FCL [18], and we have shown that the combination of FCL and the compliance-by-design methodology does not suffer from the problem of not being able to correctly asses the compliance of a business process affecting compliance frameworks (e.g., MoBuCom [26], COMPAS [27]) based on temporal logics [9]. Furthermore, Regorous was successfully tested for real industry scale compliance problems.

Acknowledgment

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

References

[1] G. Governatori and S. Sadiq, “The journey to business process compliance”, in *Handbook of Research on BPM*, IGI Global, 2009, ch. 20, pp. 426–454.

[2] G. Governatori, Z. Milosevic and S. Sadiq, “Compliance checking between business processes and business contracts”, in *Proc. EDOC 2006*, IEEE Computing Society, 2006, pp. 221–232.

[3] L. T. Ly, F. M. Maggi, M. Montali, S. Riderle-Ma and W. M. P. van der Aalst, “Compliance monitoring in business processes: Functionalities, application, and tool-support”, *Information Systems*, 2015.

[4] J. Becker, P. Delfmann, M. Eggert and S. Schwittay, “Generalizability and applicability of model-based business process compliance-checking approaches – a state-of-the-art analysis and research roadmap”, *BuR – Business Research Journal*, vol. 5, no. 2, pp. 221–247, 2012.

[5] G. Governatori, “Business process compliance: An abstract normative framework”, *IT – Information Technology*, vol. 55, no. 6, pp. 231–238, 2013.

[6] M. Hashmi, G. Governatori and M. T. Wynn, “Normative requirements for regulatory compliance: An abstract formal framework”, *Information Systems Frontiers*, 2015.

[7] ——, “Modeling obligations with event-calculus”, in *Proc. RuleML 2014*, ser. LNCS, vol. 8620, Springer, 2014, pp. 296–310.

[8] G. Governatori, “Thou shalt is not you will”, in *Proc. ICAIL 2015*, ACM, 2015, pp. 63–68.

[9] G. Governatori and M. Hashmi, “No time for compliance”, in *Proc. EDOC 2015*, IEEE, 2015.

[10] S. Sadiq and G. Governatori, “Managing regulatory compliance in business processes”, in *Handbook of Business Process Management*, vol. 2, Springer, 2010, ch. 8, pp. 157–173.

[11] M. Dumas, M. La Rosa, J. Mendling and H. A. Reijers, *Fundamentals of Business Process Management.* Springer, 2013.

[12] M. Hashmi, G. Governatori and M. T. Wynn, “Business process data compliance”, in *Proc. RuleML 2012*, ser. LNCS, vol. 7438, Springer, 2012, pp. 32–46.

[13] S. Sadiq, G. Governatori and K. Naimiri, “Modelling of control objectives for business process compliance”, in *Proc. BPM 2007*, ser. LNCS, Springer, 2007, pp. 149–164.

[14] T. F. Gordon, G. Governatori and A. Rotolo, “Rules and norms: Requirements for rule interchange languages in the legal domain”, in *Proc. RuleML 2009*, ser. LNCS, Springer, 2009, pp. 282–296.

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

[16] G. Governatori and Z. Milosevic, “Dealing with contract violations: Formalism and domain specific language”, in *Proc. EDOC 2005*, IEEE Computer Society, 2005, pp. 46–57.

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

[18] G. Governatori and A. Rotolo, “A conceptually rich model of business process compliance”, in *Proc. APCCM 2010*, ser. CRPIT, vol. 110, ACS, 2010, pp. 3–12.

[19] G. Antoniou, D. Billington, G. Governatori and M. J. Maher, “Representation results for defeasible logic”, *ACM Transactions on Computational Logic*, vol. 2, no. 2, pp. 255–287, 2001.

[20] 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.

[21] G. Governatori, F. Olivieri, A. Rotolo and S. Scannapieco, “Computing strong and weak permissions in defeasible logic”, *Journal of Philosophical Logic*, vol. 42, no. 6, pp. 799–829, 2013.

[22] G. Governatori and A. Rotolo, “An algorithm for business process compliance”, in *Proc. JURIX 2008*, ser. Frontieres in Artificial Intelligence and Applications, vol. 189, IOS Press, 2008, pp. 186–191.

[23] S. Colombo Tosatto, G. Governatori and P. Kelsen, “Business process regulatory compliance is hard”, *IEEE Transactions on Services Computing*, 2014, online first.

[24] H.-P. Lam and G. Governatori, “The making of SPINdle”, in *Proc. RuleML 2009*, ser. LNCS, Springer, 2009, pp. 315–322.

[25] G. Governatori and S. Shek, “Regorous: A business process compliance checker”, in *Proc. ICAIL 2013*, ACM, 2013, pp. 245–246.

[26] F. Maggi, M. Montali, M. Westergaard and W. van der Aalst, “Monitoring business constraints with linear temporal logic: an approach based on colored automata”, in *Proc. BPM 2011*, ser. LNCS, vol. 6896, Springer, 2011, pp. 132–147.

[27] A. Elgammal, O. Türetken and W.-J. van den Heuvel, “Using patterns for the analysis and resolution of compliance violations”, *International Journal of Cooperative Information Systems*, vol. 21, no. 1, pp. 31–54, 2012.