An Abstract Normative Framework for Business Process Compliance

Guido Governatori

Abstract
In this paper we propose an abstract framework to model the deontic notions relevant for business process compliance. In particular, we provide a comprehensive classification of the obligation types relevant for modelling whether a process is compliant, and we describe their semantics in terms of execution traces.

Keywords
Regulatory Compliance, Business Processes, Deontic Logic

NICTA, Queensland Research Laboratory

1. Introduction

The study of IT techniques to support compliance is gaining momentum in the area of Enterprise Information Systems. The number and complexity of compliance initiatives and frameworks is growing and more and more businesses are required to provide compliance certification by such frameworks. In the past few years several research approaches have been proposed (see [1, 2] for comprehensive lists of such approaches).

Regulatory compliance is defined as the set of actives and policies in places in an enterprise to ensure the business activities required to achieve the business goals of the company comply with the relevant normative requirements. Here *normative requirements* must be understood with a very broad interpretation. They include statutory laws, regulations, industry codes, standards, internal policies, ... Despite the differences, they share a common aspect: they describe the obligations, prohibitions and permissions an organization is subject to in its day to day business. *Business Process Compliance* has been defined as a relationship between the specifications of a process and the formal representation of the regulatory frameworks relevant of the process [8, 11].

The aim of this paper is not to provide a yet another system for compliance, but a conceptual abstract framework, independent of any language, to model normative requirements. The resulting framework can be used for several purposes: for example, to study formal properties of business process compliance.

For the purpose of this paper a process will be understood as a set of sequences of tasks. In addition every task has associated to it a set of effects where an effect is just a formula of the underlying language.

For the representation of the normative requirements, we propose a classification of the various normative positions (i.e., obligations, permissions, prohibitions) and we provide further refinements for them along various dimensions (e.g., temporal, compensability, perdurance). For each class we provide the semantics and examples, extracted from actual acts, codes and regulations, illustrating the particular types of normative positions.

2. 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). 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

where 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. 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 respecting the order given by the connectors.

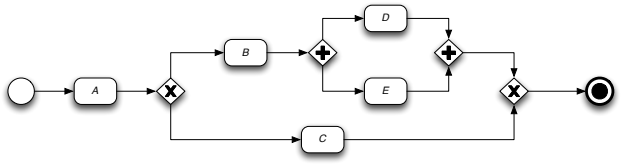


Figure 1. Example of a business process model in standard BPMN notation

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 $\langle A, C\rangle$, $\langle A, B, D, E\rangle$ and $\langle A, B, E, D\rangle$. 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) [10]. To capture this aspect [12] 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 $\mathcal{L}$ 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). 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.

3. Business Process Compliance

The set of traces of a given business process describes the behavior 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 normative system[1] if it does not breach the normative system. Given that, in general, it is possible to perform a business process in many different ways, thus we can have two notions of compliance, namely:

(S1) A process is (fully) compliant with a normative system if it is impossible to violate the normative system while executing the process.

The intuition about the above condition is that no matter in which way the process is executed, its execution does not violate the normative system. For the second one we consider the case that there is an execution of the process that does not violate the norms.

(S2) A process is (partially) compliant with a normative system if it is possible to execute the process without violating the normative system.

Based on the above intuition we can give the following definition:

Definition 1 *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 $\mathcal{N}$.*

Notice that in (S1) and (S2) compliance means "lack of violations" while in Definition 1 we had "comply with". For the purpose of this paper we will treat these two concepts as equivalent. More precisely they are related by the following definition.

Definition 2 *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.*

In Section 4 we are going to introduce various types of norms. For each type we are going to describe its 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 2. Weak compliance is defined as follows:

Definition 3 *A trace t is weakly compliant with a normative system $\mathcal{N}$ iff every violated norm has been compensated for.*

Remark 1 *It is not the scope of this paper to describe how the sequences of states corresponding to the executions of a process are obtained. The task of specifying how the function State is implemented is left to specific compliance applications.*

4. 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.

[1] Here, by normative system we simply mean a set of norms, where a norm is a formula in the underlying (deontic) language. For a business process the normative system could vary from a particular regulation, to a specific statutory act, a set of best practices, a standard, simply a policy internal to an organisation or a combination of these types of prescriptive documents.

A comprehensive list of normative effects is provided in [3]. 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*.[2]

Let us start by consider the basic definitions for such concepts:[3]

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 or the prohibition to the contrary does 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 can be compensated for, and processes with compensated violations are still compliant [5, 8]; for example contracts typically contain compensatory clauses specifying penalties and other sanctions triggered by breaches of other contract clauses [4]. 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 [13]. 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 fulfill 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 fulfill 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 fulfill 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. In the next section we will provide formal definitions for the notions introduced in this section and for each case we will show examples taken form statutory Acts and other legally binding documents.

5. Modelling Obligations

In this section we provide the formal definitions underpinning the notion of compliance. In particular we formally define the different types of obligations introduced in Section 4.

Definition 4 (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 definition specifying when the process has to

[2] There are other deontic effects, but these can be derived from the basic ones, see [13].

[3] 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`.

fulfill the various obligations (depending on their type) to be deemed compliant.

Remark 2 *Similarly to Remark 1 we are not interested in the mechanisms that establish which obligations are in force and when. This is the scope of specific compliance applications or implementations.*

Definition 5 (Punctual Obligation) *Given a process p and a trace $t \in \mathcal{T}_P$, an obligation o is a* punctual obligation *in t if and only if $\exists n \in \mathbb{N}$ such that*

1. $o \notin Force(t, n-1)$,
2. $o \notin Force(t, n+1)$, *and*
3. $o \in Force(t, n)$.

A punctual obligation o is violated *in t if and only if $o \notin State(t, n)$.*[4]

A punctual obligation is an obligation that is in force in one task of a trace (it might be the case that there are multiple instances in which the obligation is in force). The obligation is violated if what the obligation prescribes is not achieved in or done by the task, where this is represented by the literal not being in the set of literals associated to the task in the trace.

Definition 6 (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 $\forall k\colon n \leq k \leq m,\ o \notin State(t, k)$.*

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.

Example 1 *Australian Telecommunications Consumers Protection Code 2012 (TCPC 2012). Article 8.2.1.*
A Supplier must take the following actions to enable this outcome:

(a) ***Demonstrate fairness, courtesy, objectivity and efficiency:*** *Suppliers must demonstrate, fairness and courtesy, objectivity, and efficiency by:*
  (i) Acknowledging a Complaint:
    A. immediately where the Complaint is made in person or by telephone;
    B. within 2 Working Days of receipt where the Complaint is made by email; ....

The obligation to acknowledge a compliant made in person or by phone (8.2.1.a.i.A) is a punctual obligation, since it has to be done 'immediately' while receiving it (thus it can be one of the activities done in the task 'receive complaint'). 8.2.1.a.i.B on the other hand is an achievement obligation since the clause gives a deadline to achieve it. In addition it is a non-preemptive obligation. It is not possible to acknowledge a complaint before having it.

Example 2 *Anti-Money Laundering and Counter-Terrorism Financing Act 2006. Clause 54 (Timing of reports about physical currency movements).*

(1) A report under section 53 must be given:
  (a) if the movement of the physical currency is to be effected by a person bringing the physical currency into Australia with the person—at the time worked out under subsection (2); or
  [...]
  (d) in any other case—at any time before the movement of the physical currency takes place.

Clause (d) illustrates a preemptive obligation. The obligation is in force when a financial transaction occurs, and the clause explicitly requires the report to be submitted to the relevant authority before *the transaction actually occurs (it might be the case that the transaction never occurred).*

Definition 7 (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.

Example 3 *TCPC 2012. Article 8.2.1.*
A Supplier must take the following actions to enable this outcome:

(v) not taking Credit Management action in relation to a specified disputed amount that is the subject of an unresolved Complaint in circumstances where the Supplier is aware that the Complaint has not been Resolved to the satisfaction of the Consumer and is being investigated by the Supplier, the TIO or a relevant recognised third party;

In this example, as it is often the case, a maintenance obligation implements a prohibition. Specifically, it describes the

[4] For the conditions defining when an obligation is violated we assume the same conditions defining the type of the obligation. For example, in this case $\exists n \in \mathbb{N}$ such that $o \in Force(t, n)$.

prohibition to initiate a particular type of activity until either a particular event takes place or a state is reached.

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 8 and 9). 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 10).

Definition 8 (Compensation) *A* compensation *is a function* $Comp\colon \mathcal{L} \mapsto 2^{\mathcal{L}}$.

Definition 9 (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 10 (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 2. from the above condition.

Compensations can be used for two purposes. The first is to specify alternative, less ideal outcomes. The second is to capture sanctions and penalties. Examples 4 and 5 below illustrate, respectively, these two usages.

Example 4 *TCPC 2012. Article 8.1.1.*
A Supplier must take the following actions to enable this outcome:

(a) ***Implement a process****: implement, operate and comply with a Complaint handling process that:*
  (vii) *requires all Complaints to be:*
    A. *Resolved in an objective, efficient and fair manner; and*
    B. *escalated and managed under the Supplier's internal escalation process if requested by the Consumer or a former Customer.*

Example 5 *YAWL Deed of Assignment, Clause 5.2.*[5]
Each Contributor indemnifies and will defend the Foundation against any claim, liability, loss, damages, cost and expenses suffered or incurred by the Foundation as a result of any breach of the warranties given by the Contributor under ***clause 5.1****.*

[5] `http://www.yawlfoundation.org/files/YAWLDeedOfAssignmentTemplate.pdf`, retrieved on March 28, 2013.

The final definition is that of perdurant obligation. The intuition behind it is that there is a deadline by when the obligation has to be fulfilled. If it is not fulfilled by the deadline then a violation is raised, but the obligation is still in force. Typically, the violation of a perdurant obligation triggers a penalty, thus if the perdurant obligation is not fulfilled in time, then the process has to account for the original obligation as well as the penalties associated with the violation.

Definition 11 (Perdurant Obligation) *Given a process* P *and a trace* $t \in \mathcal{T}_P$*, an obligation* o *is a* perdurant 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. $\forall k\colon n \leq k \leq m, o \in Force(t, k)$.

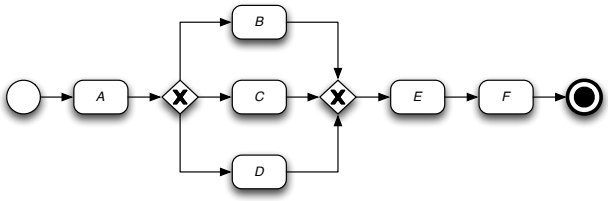
A perdurant obligation o *is* violated *in* t *if and only if*

$$\exists k\colon n < k < m,\ \forall j, j \leq k,\ o \notin State(t, j)$$

Consider again Example 1. Clauses TCPC 8.2.1.a.i.A and 8.2.1.a.i.B state what are the deadlines to acknowledge a complaint, but 8.2.1.a.i prescribes that complaints have to be acknowledged. Thus, if a complaint is not acknowledged within the prescribed time then either clause A or B are violated, but the supplier still has the obligation to acknowledge the complaint. Thus the obligation in clause (i) is a perdurant obligation.

6. Example

In this section we introduce a synthetic example illustrating some of the notions defined in the previous section.



The process P above generates the following set of traces:

$$\mathcal{T}_P = \{t_1 = \langle A, B, E, F\rangle, t_2 = \langle A, C, E, F\rangle, t_3 = \langle A, D, E, F\rangle\}$$

Each task in P is annotated with a set of effects. We use $ann(X)$ for the set of effects of task X. More specifically for each task X in the process with the exception of task D, $ann(X) = \{x\}$, while for D, $ann(D) = \{d, \neg a\}$.

The state reached after the execution of a task is computed using by cumulating the effects of the task based on an update semantics (in case of a conflict between one effect from a previous task and an effect of the current task, the old one is removed), namelly:

$$State(t, n+1) = (State(t, n) \setminus \{\neg c\colon c \in ann(n+1)\}) \cup ann(n+1)$$

The sequences of states corresponding to the traces of P are:

$$t_1 : \langle\{a\},\{a,b\},\{a,b,e\},\{a,b,e,f\}\rangle$$
$$t_2 : \langle\{a\},\{a,c\},\{a,c,e\},\{a,c,e,f\}\rangle$$
$$t_3 : \langle\{a\},\{\neg a,d\},\{\neg a,d,e\},\{\neg a,d,e,f\}\rangle$$

The mechanism to determine the obligations in force for a particular task (in a trace) is described by the following principles: if $o \in Force(t, n+1)$, then

1. there is a norm $n \in \mathcal{N}$ with the form "if c then it is obligatory o before e" such that: $State(t,n) \Vdash c$ and $State(t,n-1) \nVdash c$; or
2. $o \in Comp(x)$ and $State(t,n+1) \Vdash violation(x)$; or
3. $o \in Force(t,n)$ and either
   (a) $State(t,n) \nVdash e$; or
   (b) $o \notin State(t,n)$ where o is an achievement obligation; or
   (c) for every norm $m \in \mathcal{N}$ with the form "if c then it is permitted to skip o", $State(t,n) \nVdash c$.

In the clauses above we used $State(t,n) \Vdash c$ to denote that the condition c is satisfied by the state at the n-th task in trace t. c can be a (Boolean) combination of literals: effects of tasks or conditions about compliance (e.g., obligations in force, violations, ...).

Clauses 1 and 2 above give the conditions to determine that an obligation enters into force. In the first case there is a norm that becomes applicable at task n, and the obligation is in force at task $n+1$. In the second case, the obligation is in force because it is a compensation of a violation, and the violation has been detected in the current task (i.e., $n+1$-th task). Clause 3, on the other hand, gives the conditions to specify that an obligation is no longer in force. Condition (b) allows us to remove an achievement obligation from the set of active obligations when the obligation has been fulfilled. Condition (c) states that there is a derogatory norm that permits that the obligation is no longer the case (a more natural reading would be that the opposite is permitted). Condition (a) indicates that an obligation is no longer active when we reach its deadline. Notice, however, that a deadline for a perdurant obligation is just to signal that there is a violation and does not remove the obligation itself.

Suppose that the process is governed by a normative system $\mathcal{N}$ with the following norms:

(N1) If a then it is obligatory to have b before e.
(N2) If b is not the case then it is obligatory to have f.
(N3) If d then it is permitted to skip b.

The set of obligations in force for the processes are (assuming that b is an achievement obligation)

$$t_1 : \langle\emptyset,\{b\},\emptyset,\emptyset\rangle$$
$$t_2 : \langle\emptyset,\{b\},\{b\},\{f\}\rangle$$
$$t_2' : \langle\emptyset,\{b\},\{b\},\{b,f\}\rangle)$$
$$t_3 : \langle\emptyset,\{b\},\emptyset,\emptyset\rangle$$

We have included two instances of trace t_2, the first one where b is non-perdurant, and t_2' when b is perdurant. At this stage we can combine the information about the sequences of states and the sequences obligation in force to determine whether the process is compliant or not. Traces t_1 and t_3 are both compliant, but for different reasons. t_1 because the obligation of b has been fulfilled: task A makes b obligatory, and the effect of task B is b, thus the obligation of b has been discharged. On the contrary, we do not achieve b in trace t_3, but the performance of task D renders b no longer obligatory, and thus there is no violation is b is not present. Trace t_2 is only weakly compliant. Similarly to the other traces the obligation of b is in force, but in this case we reach task E where e holds without discharging this obligation. However, norm N2 specifies that $Comp(b) = \{f\}$, thus, after E the only obligation in force is f which is achieved at the next step, compensation the violation of b. If we consider t_2' instead, then b perdures after E, and then the obligations in force are both b and f, making trace t_2' non-compliant.

7. Conclusion

In this paper we presented an abstract framework to describe the key concepts of business process compliance. In particular we provided a comprehensive classification of obligations and their semantics in terms of the execution traces of a process. The proposed model is neutral from specific logics for reasoning with norms and process model formalisms.

To the best of our knowledge PCL (Process Compliance Logic) [6, 7] is the only compliance management framework supporting all the types of normative requirements presented in this paper. PCL and the abstract framework developed in this paper have been evaluated with an industry study reported in [9]. The study examined Section 8 of the 2012 Australian Telecommunication Customer Protection Code about complaint handling, where all types of normative requirement described above were found. In addition the compliant handling process of an industry partner operating in the sector where model and tested for compliance. The exercise was fruitful insofar as the industry partner was able to identify some non compliance issues with the novel code, and consequently was able to rectify them, and to generate compliance verifiable processes.

Acknowledgment

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

References

[1] 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.* 5(2): 221–247, 2012.

[2] M. El Kharbili. "Business Process Regulatory Compliance Management Solution Frameworks: A Comparative Evaluation". In *APCCM 2012*. CRPIT 130, pp. 23–32, 2012.

[3] T. F. Gordon, G. Governatori, and A. Rotolo. "Rules and Norms: Requirements for Rule Interchange Languages in the Legal Domain". In *RuleML 2009*. LNCS 5858, pp. 282–296. Springer, 2009.

[4] G. Governatori. "Representing Business Contracts in RuleML". *International Journal of Cooperative Information Systems*. 14(2-3): 181–216, 2005.

[5] G. Governatori and Z. Milosevic. "Dealing with contract violations: formalism and domain specific language". In *EDOC 2005*, pp. 46–57. IEEE Computer Society, 2005.

[6] G. Governatori and A. Rotolo. "A Conceptually Rich Model of Business Process Compliance". In *APCCM'10*. CRPIT 110, pp. 3–12, 2010.

[7] G. Governatori and A. Rotolo. "Norm Compliance in Business Process Modeling". In *RuleML'10*. LNCS 6403, pp. 194–209. Springer, Heidelberg, 2010.

[8] G. Governatori and S. Sadiq. "The Journey to Business Process Compliance". In *Handbook of Research on Business Process Management*, pp. 426–454. IGI Global, 2009.

[9] G. Governatori and S. Shek. "Rule Based Business Process Compliance". In *RuleML2012@ECAI Challenge*. CEUR Workshop Proceedings 874, article 5, 2012.

[10] M. Hashmi, G. Governatori, and M. T. Wynn. "Business Process Data Compliance". In *RuleML 2012*. LNCS 7438, pp. 32–46. Springer, Berlin, 2012.

[11] S. Sadiq and G. Governatori. "Managing Regulatory Compliance in Business Processes". In van Brocke, J., and M. Rosemann, eds. *Handbook of Business Process Management*. Vol. 2, pp. 157–173. Springer, Berlin, 2010.

[12] S. Sadiq, G. Governatori, and K. Namiri. "Modeling Control Objectives for Business Process Compliance". In *BPM 2007*. LNCS 4714, pp. 149–164. Springer, Berlin, 2007.

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