Modeling Obligations with Event-Calculus *

Mustafa Hashmi[1,2], Guido Governatori[1,2] and Moe Thandar Wynn[2,1]

[1] NICTA, Queensland Research Laboratory, 2 George St. Brisbane Australia
{mustafa.hashmi,guido.governatori}@nicta.com.au
[2] Queensland University of Technology (QUT) Brisbane, Australia
m.wynn@qut.edu.au

Abstract. Time plays an important role in norms. In this paper we start from our previously proposed classification of obligations, and point out some shortcomings of Event Calculus (EC) to represent obligations. We propose an extension of EC that avoids such shortcomings and we show how to use it to model the various types of obligations.

Keywords: Legal norms, Event Calculus, Temporal aspect, Compliance

1 Introduction

Time plays an essential role in norms, legal reasoning and in areas governed by norms. For example many of the normative requirements in the area of business process compliance concern the temporal aspects of norms. Suppose you have a contract specifying that one party has thirty days to pay for an invoice, and that goods cannot be delivered without payment. Thus you have an obligation to pay after receiving an invoice, which, in turn, requires that the payment *must be made before* the time of delivery. Receiving the invoice triggers (enforces) the obligation to make a payment to complete the transaction. Accordingly we have conditions that must be fulfilled in a determined time interval or within a given deadline, and other conditions that must happen before or after specific events. Moreover, some obligations may include conditions that *must persist over an interval of time* e.g., continuous monitoring of the patient's blood pressure and ECG during a surgical operation. Regardless of the type, validity and nature of the legal effect(s) that an obligation represents, the temporal aspect of an obligation revolves around the following generic aspects [17]: (i) *the time when an obligation is in force*, (ii) *the time when an obligation is fulfilled*, and (iii) *the time of application*. Accordingly, when a business process is subject to norms, it is particularly important that the process complies with the obligations imposed by the norms for the whole duration of its validity and meets the deadlines, and follows constraints for maintaining and delaying actions.

Capturing the real meaning of norms is paramount for modelling and reasoning about compliance checking of business processes, and, in general, for legal reasoning. It is also important that the chosen language supports the highest degree of abstraction to model the real meaning of the norms and the obligations they define: this means it should be able to model states of affairs, actions as well as (temporal) relationships

*NICTA is funded by the Australian Government through the Department of Communication and the Australian Research Council through the ICT Center of Excellence Program.

A. Bikakis, P. Fodor and D. Romanu (eds), *RuleML 2014*, LNCS 8620, pp. 296–306

between activities. Many studies have been conducted for modelling obligations, and various classifications of obligations have been identified in these studies, in particular in the context of business process compliance where *time* is the key concept of such classifications, see among others [13,10,8]. For example, [19] classifies obligations from the legal viewpoint while [13] classifies obligations along the temporal structure and the temporal distribution of the obligations. [8] characterises the types of obligations based on deadlines, and [3] classifies obligations types as existence, choice, relation, and negative constraints. These classifications do not encompass various types of obligations based on the time, effects of an obligation on other obligations and obligations arising from the violations. In [7,12] we provided a classification of obligations along temporal dimensions. The key aspects of the classification are: what constitutes the violation in terms of the temporal validity of an obligation, and whether violated obligations can be compensated for or not. In the classification, along the temporal dimension, for each type of obligations we specified when an obligation comes into force and until when it remains in force or it is violated at a particular time point. Unlike other classifications, our proposed classification encompasses the generic temporal model about the validity and persistence effects of obligations after violations. Given our new classification, the natural question is *how to model each element of the classification of obligations for business process compliance checking*.

The families of Deontic Logics (DL), Temporal Logics (TL), and EventCalculus (EC) are widely used formalisms for modeling norms. Each of these formalisms has a reasonable degree of expressiveness to model different types of obligations yet they have limitations. Our starting point to model norms, in particular the new classes of obligations, is the classical EC [14] because it provides a logical framework for representing and modeling the effects of events and the current state of affairs in terms of fluents. Also, it has the ability to model the time when fluents come to existence and cease to hold dynamically [5]. One may argue that modeling the deontic notions with EC is rather well developed as several variants of EC already exist (see, [16,18] for further listing of EC variants), and widely used for reasoning and representing the legal knowledge (see, Section 6 for a detailed discussion on some such approaches), but we believe that the EC has some major issues for reasoning about legal norms. One of such issues is related to the basic predicate of EC $Initiates(E,X,T)$. Its meaning is that event E at time T initiates the fluent X, and the fluent holds from the next instant of time (see Section 3 and Axiom A1 below for the details). This effectively means that the norm enters into force at the next instant. However, for legal norms, this might not be the case. There are cases where the norm enters into force at the same instant as the triggering event happens e.g., the obligation to remove shoes when one enters in a mosque or the norms is in force after a delay e.g., a complaint cannot be acknowledged until all details pertaining is issue have been received.[3]

In the context of business process compliance checking, the aim of this paper is to explore whether or not the different obligation classes defined in our classification model can be faithfully represented using the discrete event driven formalism the EC.

[3]In addition it is possible to have that a norm enters in force retroactively. Thus the fluent holds before the event that initiates it. We blatantly ignore this aspect in this paper.

The paper is structured as follows: in Section 2 we revisit the classification of normative requirements proposed elsewhere ([7,12]) and provide formal definitions of the concepts. Section 3 provides a terse background of the EC and introduces new predicates for modeling the legal norms followed by the modeling of various obligation types using the new predicates in Section 4. The proof sketch of the provided axioms is given in Section 5 followed by a short discussion on related studies in the problem domain in Section 6. Section 7 concludes the paper with some final remarks.

2 Normative Requirements Revisited

The purpose of this section is to provide a summary of the notions and the classes of obligations defined in our classificatory model. For more detailed discussions and concrete examples of the various types of obligations taken from real legal acts, see [7,12]. The definitions below also provide precise semantics of these notions and they will be used to evaluate our proposed extension to EC.

Norms regulate the behaviour of their subjects and produce normative effects when applied. From a business process compliance perspective the normative effects of interest are the deontic effects. The three basic deontic effects –from which other deontic effects can be derived (see, [19])– are: *obligation*, *prohibition*, and *permission*.

An *Obligation*[4] is a situation, act or a course of actions one is legally bound to and if it is not achieved or performed results in a violation; whereas for *prohibition*, one should avoid a certain course of actions to avoid a violation. *Obligations* and *prohibitions* are constraints that limit the behaviour of a business process; and both types can be violated. Notice that a prohibition is a negative obligation (i.e., obligation not), thus, when we speak of obligations we include prohibitions as well. *Permissions*, on the other hand, are constraints that cannot be violated thus they do not play a direct role in compliance. Instead, they can be used to determine that there are no obligations or prohibitions to the contrary.

Compliance means to identify whether a business process violated a set of obligations. Thus the first step is to determine *whether* and *when* an obligation is in force. Essentially, a norm can specify when an obligation is in force at a particular time point only (*non-persistent obligations*), or more often, a norm indicates when an obligation enters into force. An obligation remains in force until it is terminated or removed (*persistent obligations*).

Non-Persistent obligations are also called *punctual obligations*: the obligation contents are immediately achieved otherwise a violation is triggered. In contrast, a *persistent obligation* which is to be obeyed for all time instances within the interval it is in force is a *maintenance obligation*. If achieving the contents of an obligation at least once is enough, then it is an *achievement obligation*. For an achievement obligation, if the obligation could be fulfilled even before it is actually in force, we speak of a *preemptive* obligation; otherwise it is a *non-preemptive* obligation.

An important aspect of obligations that differentiates them from other types of constraints is that an obligation can be violated. However, the violation of an obligation

[4]The definition is taken from the glossary created by the OASIS LegalRuleML workgroup `http://www.oasis-open.org/apps/org/workgroup/legalruleml`

does not necessarily mean the termination of interaction of a business process because some violations can be compensated for while keeping the underlying process still compliant [9,11]. However, not all violations are compensable, and an uncompensated violation would mean the process is non-complaint. If an obligation persists after being violated, it is a *perdurant obligation* if not then we have a *non-perdurant obligation*.

Next we formally define the meanings of the obligations, all we need is the concept of timeline, i.e., a (possibly infinite[5]) totally ordered discrete set of time points. Also, we assume that the timeline has a minimum. In what follows, we assume the existence of a logical language $\mathfrak{L}$ (can be a set of atomic propositions) on which the formulas are written to model obligations and the representation of the environment.

Definition 1 (State). *Given a timeline, we define a function State*: $\mathbb{N} \mapsto 2^{\mathfrak{L}}$.

The meaning of the function *State* is to identify what formulas are evaluated as true at the n-th time instant of a timeline.

Definition 2 (Obligation in Force). *Given a timeline, we define a function Force*: $\mathbb{N} \mapsto 2^{\mathfrak{L}}$.

The meaning of the function *Force* is to identify the obligations in force at the n-th instant of time in a given timeline.

Definition 3 (Punctual Obligation). *Given a timeline, an obligation o is a punctual obligation if and only if:*

$$\exists n \in \mathbb{N} : o \notin Force(n-1), o \notin Force(n+1), o \in Force(n)$$

A punctual obligation is violated at n if and only if $o \notin State(n)$.

The conditions of a *punctual obligation* must be fulfilled immediately otherwise we have a violation i.e., o is violated at time n if o is not true at n (or at the n-th instant of time in the timeline).

Definition 4 (Persistent Obligation). *Given a timeline, an obligation o is a persistent obligation if and only if:*

$$\exists n, m \in \mathbb{N} : n < m, o \notin Force(n-1), o \notin Force(m+1),$$
$$\forall k : n \leq k \leq m, o \in Force(k)$$

The obligation o is in force *between n and m.*

A *persistent obligation* is an obligation in force in an interval time, and can be further classified as: (a) *achievement*, and (b) *maintenance* obligation. The violation conditions for a persistent obligation can be derived from the violation conditions of these subclasses.

Definition 5 (Achievement Obligation). *Given a timeline, an obligation o is an achievement obligation if and only if $\exists n, m \in \mathbb{N}$, $n < m$ such that o is a persistent obligation in force between n and m.*

An achievement obligation o in force between n and m is violated if and only if:

[5]Notice an infinite timeline is isomorphic to the set of natural numbers (and we can restrict to a finite set of natural numbers in case of a finite timeline).

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

An *achievement obligation* is in force in an interval in the timeline, and can be further classified as: *preemptive* and *non-preemptive*. A preemptive achievement obligation o is an obligation that can be fulfilled even before the obligation is actually in force. In contrast, a non-preemptive achievement obligation can be discharged only after it enters in force. The violation of an achievement obligation depends on whether we have a preemptive or non-preemptive obligation. Notice that the violation of an achievement obligation can only be asserted after the deadline.

Definition 6 (Maintenance Obligation). *Given a timeline, an obligation o is a maintenance obligation if and only if* $\exists n, m \in \mathbb{N}, n < m$ *such that o is a persistent obligation in force between n and m.*

A maintenance obligation o in force between n and m is violated if and only if

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

Unlike achievement obligations, a *maintenance obligation* must be complied with for all the instances between the interval otherwise we have a violation. Also, no deadline is required for a maintenance obligation insofar we do not need it to detect a violation. The deadline signal that after that instant the obligation is no longer in force. Furthermore it is possible to define maintenance obligation without a deadline, meaning the that the obligation remains in force forever after its activation; for this case, one has to drop the reference to instant m in the above definition.

The next three definitions capture the notion of compensation of a violation. A compensation is a set of obligations that are in force after a violation of an obligation, and fulfilling them makes amend for the violation.

Definition 7 (Compensation). *A compensation is a function* $Comp\colon \mathfrak{L} \mapsto 2^{\mathfrak{L}}$.

The intuition behind the function *Comp* is that it associates to each formula a set of formulas, meaning that if a formula corresponds to an obligation, and the obligation is violated, then the violation is compensated (or excused) by the formulas associated to the obligation. This is formalised by the next definition.

Definition 8 (Compensable). *Given a timeline, an obligation o is compensable if and only if* $Comp(o) \neq \emptyset$ *and* $\forall o' \in Comp(o), \exists n \in \mathbb{N} : o' \in Force(n)$.

Notice that we have two requirements for an obligation to be compensable: the first is that there are ways to make amend i.e., that $Comp \neq \emptyset$, and the second is that the actions that compensate are recognised as such (they are obligations in force) or they are not forbidden. Finally, in the most general form, there are no temporal requirements on when the compensation happens.[6]

Since the compensations are obligations themselves they can be further violated, accordingly they can be compensated for the violations as well, thus a recursive definition of a compensated obligation is required.

[6] In vast majority of cases, it is expected that the compensatory obligations are in force after the violation. However, the definition above does not exclude retroactive compensations.

Definition 9 (Compensated Obligation). *Given a timeline, an obligation o is compensated if and only if it is violated and for every* $o' \in Comp(o)$ *either: 1.* o' *is not violated; 2.* o' *is compensated.*

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.

The last type of obligation is that of *perdurant obligation*. The idea is that when an obligation is violated, the violated obligation is not terminated yet remains in force. Given the conditions of primary obligation an obligation may perdure no matter how many times the obligation has been violated. The violation of a perdurant obligation results in penalty for which one has to consider the original obligation as well as penalties associated with the violation.

Definition 10 (Perdurant). *Given a timeline, an obligation o is a perdurant obligation with a deadline d if and only if o is in force between n and m, and* $n < d < m$.

A perdurant obligation o with a deadline d in force between n and m is violated if and only if

$$\forall j, j \leq d, o \notin State(j)$$

3 Event Calculus

The Event Calculus [14] is a well known event based formalism for reasoning about '*events and change*' and the '*effects of change*' resulting from the occurrence of events over time. EC provides a set of rich axioms for capturing the behaviour of dynamic occurrences of both domain dependent and domain independent events. Hence the formalism is particularly suitable to model the behaviour of a variety of dynamic systems. It is based on the idea of the state that time-varying properties of the world, called fluents hold at particular time-points initiated by some event at an earlier time, and not terminated by some other event between that time period. Accordingly, a fluent does not hold at some time if it was previously terminated and not resumed during that time [15]. In contrast, domain dependent axioms illustrate the situations under which an event initiates and terminates. In this paper, we make use of the predicates and axioms depicted in Table **??** from [16]. The language provides predicates expressing the various states of an event occurrence, e.g., *Happens* (occurrence of an event at a time point), *Initiates* (an event triggers the property of the system), *Terminates* (an event terminates the property of the system), and *HoldsAt* (that the property of the system holds at a point of time). In addition, some auxiliary predicates to express premature termination (*Clipped*) and resumption (*Declipped*) of a fluent at a particular point of time between the time interval are given. The *InitiallyTrue* and *InitiallyFalse* allow for the modeling of system's state where only partial information about the domain is available. In contrast, the domain independent axioms describe the states when a fluent holds or does not hold at particular point of time.

For example, consider the following axioms [16]:

$$\begin{aligned} HoldsAt(P,T_2) \leftarrow\ & Happens(P,T_1) \wedge Initiates(X,P,T_1) \wedge \\ & \neg Clipped(T_1,P,T_2) \wedge (T_1 < T_2) \end{aligned} \tag{A1}$$

Table 1: Predicates and Axioms of the EC and meanings

Basic Predicates	
$InitiallyTrue(P)$	The fluent P is true from the beginning of time.
$InitiallyFalse(P)$	The fluent P is false from the beginning of time.
$Happens(X,T)$	Event X occurs at time T.
$Initiates(X,P,T)$	Event X initiates the variable (fluent) P at time T.
$HoldsAt(P,T)$	The variable (fluent) P holds at time T.
$Terminates(X,P,T)$	Event X terminates the variable (fluent) P at time T
Auxiliary Predicates	
$Clipped(T_1,P,T_2)$	The variable (fluent) P is interrupted sometime between T_1 and T_2.
$Declipped(T_1,P,T_2)$	The variable (fluent) P is resumed/initiated sometime between T_1 and T_2.
Domain Independent Axioms	
$HoldsAt(P,T_2) \leftarrow$	$HoldsAt(P,T_1) \wedge (T_1 < T_2) \wedge \neg Clipped(T_1,P,T_2)$
$HoldsAt(P,T_2) \leftarrow$	$Happens(P,T_1) \wedge Initiates(X,P,T_1) \wedge \neg Clipped(T_1,P,T_2) \wedge (T_1 < T_2)$
$\neg HoldsAt(P,T_2) \leftarrow$	$Happens(X,T_1) \wedge Terminates(X,P,T_1) \wedge (T_1 < T_2) \wedge \neg Declipped(T_1,P,T_2)$
$\neg HoldsAt(P,T_2) \leftarrow$	$\neg HoldsAt(P,T_1) \wedge (T_1 < T_2) \wedge \neg Declipped(T_1,P,T_2)$
$Clipped(T_1,P,T_2) \equiv$	$\exists X,T : Happens(X,T) \wedge (T_1 \leq T < T_2) \wedge Terminates(X,P,T)$
$Declipped(T_1,P,T_2) \equiv$	$\exists X,T : Happens(X,T) \wedge (T_1 \leq T < T_2) \wedge Initiates(X,P,T)$

The (Axiom A1) states that the fluent P continues to hold until an event that terminates it occurs, provided that there was an event that happened at some previous time which was a trigger for the fluent.

$$\neg HoldsAt(P,T_2) \leftarrow Happens(X,T_1) \wedge Terminates(X,P,T_1) \wedge (T_1 < T_2) \wedge \neg Declipped(T_1,P,T_2) \quad \text{(A2)}$$

Whereas (Axiom **??**) states that fluent P that has been terminated by the event X continues not to hold until it is resumed by some other event occurrence.

The above axiomatisation can be used to model the non-deterministic behaviour of a system thus EC is suitable for modeling obligations that can be effected by unpredictable situations. However, as was noted earlier in Section 1, an obligation might not enter into force immediately after the occurrence of an event rather after some time delay. A second problem is that the base predicate *Initiates* does not gurantee that the fluent in its arguments is actually *initiated* by the event. Suppose that the domain dependent axioms specify that both the events E_1 and E_2 individually initiate the fluent P, and event E_1 happens at time 10 and event E_2 at time 20, P does not holds initially and no other event initiates or terminates fluent P between 0 and 30. This means that P starts to hold from 11 and continues to hold up to 30, and event E_2 is irrelevant to determine the status of P. Also, there are cases where an obligation enters in force at the same time of initiating an event (and not the next time instant).

4 Modeling Obligations with Event Calculus

In this section we propose a set of axioms to extend the EC to model the various obligation classes of the classification model described in Section 2.

As we have seen at the end of the previous section, the standard *Initiates* and *HoldsAt* predicates of EC present some shortcomings for modelling obligations. To obviate these problems, we introduce a new 'deontically holds at' predicate $DHoldsAt(P,T)$ meaning that the 'deontic fluent', i.e., a particular type of obligation, P holds at time T. The main difference with the standard EC *HoldsAt* predicate is on the conditions of *initiation*. Each obligation has its own specific triggering events, and the happening of one of those triggering events initiates the obligation. In addition, there could be a delay (which could be null) between the time the triggering event happens and the time obligation enters in to force. A triggering event for an obligation is represented by $trigger(O^{x,T}X,N)$, where $O^{x,T}X$ is a deontic fluent, and N the delay. $O^{x,T}$ [7] represents the type of the obligation (see Section 2) and the time when the obligation enters in force T, X is a variable attached to the obligation representing the contents of the obligation, which can be either an event or a fluent, and N is the delay. As we said above the purpose of the triggering event is to initiate the obligation. For a trigger to be effective, one has to specify the conditions defining the trigger for an obligation. Also, the delay must be specified because the delay determines the difference in time from when the triggering event occurs and when the obligation enters into force.

For the termination of deontic fluents we introduce the new predicate $DTerminates(E,P,N,T_{ter})$ meaning that an event E deontically terminates the fluent P, with some delay N, at time T_{ter}. The delay N define the time distance from when the terminating event happens and the actual termination of the deontic fluent. After a deontic termination an obligation has no legal effects on the execution of the process from the time it is terminated. Also for specifying the deadlines for obligations, in the same say, we define a special deadline-triggering event $deadline(O^{x,T}X,T_d)$, where $O^{x,T}$ and X are the arguments for deadline event and serve as triggering events and T_d represents the time of the deadline event occurrence. The purpose of the deadline event is to signal the time (deadline) until when the obligation conditions must be fulfilled, a violation of the obligation conditions is triggered otherwise.

We provide generic axioms that we need to model the obligations. These axioms provide the conditions for no legal effects (not deontically Holds) after the termination of an obligation (Axiom **??**) and the conditions when no fluent deontically holds (Axiom **??**).

$$\neg DHoldsAt(X,T+1) \leftarrow \exists E : DTerminates(E,X,N,T) \tag{A3}$$

$$\neg DHoldsAt(X,T_k) \leftarrow \neg DHoldsAt(X,T) \wedge \neg Happens(trigger(X,N),T_j) \wedge (T \leq T_k) \wedge (T \leq T_j + N \leq T_k) \tag{A4}$$

In what follows we will have several cases where the trigger for an obligation does not only trigger the initiation for the obligation but also the termination. This means that we have to write expression with the following form

$$DTerminates(trigger(P,N),P,N,T) \tag{1}$$

[7] Notice $O^{x,T}$ has only one time stamp because one can be certain that an obligation holds after deontically initiated but one cannot be certain when it is going to be terminated.

where we have to repeat twice the parameters P and N. To ease readability we will use the convention of dropping the P and N from the arguments *DTerminates*, using thus

$$DTerminates(trigger(P,N),T) \tag{2}$$

The reader should keep in mind that (**??**) is a shorthand for (**??**).

4.1 Punctual obligation

The axioms describing when a punctual obligation holds are the following:

$$\begin{aligned} &DHoldsAt(O^{p,T_s}X,T_s) \leftarrow \\ &\quad \exists T_t,N : Happens(trigger(O^{p,T_s}X,N),T_t) \wedge \\ &\quad (T_s = T_t + N) \wedge N \geq 0 \end{aligned} \tag{A5}$$

$$\begin{aligned} &DTerminates(trigger(O^{p,T_s}X,N),T_s) \leftarrow \\ &\quad \exists T_t,N : Happens(trigger(O^{p,T_s}X,N),T_t) \wedge \\ &\quad (T_s = T_t + N) \wedge N \geq 0 \end{aligned} \tag{A6}$$

Let us examine in details the above axioms. An obligation is represented as a fluent; specifically the (punctual) obligation of X is represented by the fluent $O^{p,T_s}X$ where O^{p,T_s} is an obligation modality (a specific type of the obligation) and time when the obligation enters into force (T_s), and X is a variable referring the contents of obligation. In addition, we create a special event $trigger(O^xY,N)$ whose meaning is to initiate the obligation. In this way, all one has to do is to specify when an obligation enters in force by defining the conditions for the trigger. Axiom (**??**) specifies that the same event that triggers the obligation, terminates the obligation, and obligation terminates in the same time instant when it is initiated. Thus in combination with (Axiom (**??**)) we have a punctual obligation is in force for only one time instant. The axiom specifying when a punctual obligation is violated is:

$$\begin{aligned} &Happens(violation(O^{p,T_s}X),T_v) \leftarrow \\ &\quad DHoldsAt(O^{p,T_s}X,T_s) \wedge \\ &\quad \neg Happens(X,T_s) \wedge \neg HoldsAt(X,T_s) \wedge (T_v = T_s) \end{aligned} \tag{A7}$$

The violation of a punctual obligation happens when we do not have the content of the obligation at the right time. This can happen in two cases: (a) *the content is a fluent and it does not hold at the time*; or (b) *it is an event and it does not happens at the time*.[8]

Notice that we introduce a violation event ($violation(O^{p,T_v}X)$).

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;
      ....

[8] To capture that nothing is both and event and a fluent we add the axiom $\bot \leftarrow Happens(X,T) \wedge HoldsAt(X,T')$.

Consider the clause (A) of the Article 8.2.1 where the obligation must be fulfilled immediately. This can be modeled as:

$$\begin{aligned} &Happens(trigger(O^{p,T}\,Acknowledge,0),T) \leftarrow \\ &\quad Happens(Complaint,T) \wedge \\ &\quad (HoldsAt(inPerson,T) \vee HoldsAt(byPhone,T)) \end{aligned} \tag{3}$$

Suppose there is an event *Complaint* at time T and the fluent *byPhone* holds at the same time. Then from the domain Axiom (1), we derive $trigger(O^{p,T}Acknowledge,0),T)$, and then from Axioms: (**??**), (**??**) and (**??**) we obtain $DHoldsAt(O^{p,T}Acknowledge,T)$ and $\neg DHoldsAt(O^{p,T}Acknowledge,T+1)$. Meaning that the obligation to acknowledge the complaint on reception of it. Moreover, suppose that we model the acknowledgement as an event, and we have $Happens(Acknowledge,T)$, then the conditions for having a violation do not hold. Suppose now that $Happens(Acknowledge,T)$ is not true, i.e., the complaint is not acknolwedged, thus $\neg Happens(Acknowledge,T)$ is true. In addition, given that *Acknowledge* is an event, if we have $\neg HoldsAt(Acknowledge,T)$, then, we can use Axiom (**??**) to conclude that the obligation to acknowledge a complaint by phone on the spot has been violated.

4.2 Persistent Obligation

The following axiom describes a persistent obligation with a natural deadline when the fluent holds in interval:[9]

$$\begin{aligned} &DHoldsAt(O^{per,T_s}X,T_k) \leftarrow \\ &\quad \exists T_t,N : Happens(trigger(O^{per,T_s}X,N),T_t) \wedge \\ &\quad \neg DClipped(T_s,O^{per,T_s}X,T_k) \wedge \\ &\quad DTerminates(trigger(O^{per,T_s}X,N),T_e) \wedge \\ &\quad (T_s = T_t + N) \wedge (T_e > T_s) \wedge (T_s \leq T_k \leq T_e) \wedge N \geq 0 \end{aligned} \tag{A8}$$

By '*natural deadline*' we mean that if no other (relevant) event happens the obligation is in force from the T_s and T_e, and that T_e is determined by the same event that triggers the (persistent) obligation.

Achievement Obligation An achievement obligation is a special case of a persistent obligation where there might not be a natural deadline for the obligation. Hence there are two cases for achievement obligations:

(i) *when the obligation has no termination point*, i.e., initiation of achievement obligation.

$$\begin{aligned} &DHoldsAt(O^{a,T_s}X,T_s) \leftarrow \\ &\quad \exists T_t,N : Happens(trigger(O^{a,T_s}X,N),T_t) \wedge (T_s = T_t + N) \wedge N \geq 0 \end{aligned} \tag{A9}$$

(ii) *The obligation Holds at a particular time point deontically initiated and not clipped between the interval*, i.e., start time and the point until it holds.

$$\begin{aligned} &DHoldsAt(O^{a,T_s}X,T_k) \leftarrow \\ &\quad DHoldsAt(O^{a,T_s}X,T_s) \wedge \neg DClipped(T_s,O^{a,T_s}X,T_k) \wedge (T_s \leq T_k) \end{aligned} \tag{A10}$$

[9]The defintion of *DClipped* is the same as that for *Clipped* where *Terminates* is replaced by *DTerminates*.

There are two cases of the termination of an achievement obligation:

1. An arbitrary event terminates the obligation when the obligation conditions are fulfilled before the deadline of obligation.

$$\begin{aligned} &DTerminates(\_, O^{a,T_s} X, N, T_k) \leftarrow \\ &\quad Happens(\_, T_k) \wedge DHoldsAt(O^{a,T_s} X, T_k) \wedge \\ &\quad (Happens(X, T_k) \vee HoldsAt(X, T_k)) \wedge \\ &\quad FulfillTerminable(O^{a,T_s} X) \wedge (T_s \leq T_k) \end{aligned} \tag{A11}$$

The symbol '_' represents an arbitrary event, which can be anything, e.g., a new obligation, an activity or even a deadline etc., that terminates the obligation.

2. Where the deadline itself terminates the obligation.

$$\begin{aligned} &DTerminates(deadline(O^{a,T_s} X, T_d), T_d) \leftarrow \\ &\quad Happens(deadline(O^{a,T_s} X), T_d) \wedge (T_s \leq T_d) \end{aligned} \tag{A12}$$

The axiom for the termination of a preemptive obligation is:

$$\begin{aligned} &DTerminates(\_, O^{a,T_s} X, N, T_e) \leftarrow \\ &\quad Happens(\_, T_e) \wedge DHoldsAt(O^{a,T_s} X, T_s) \wedge \\ &\quad \exists T' : (Happens(X, T') \vee HoldsAt(X, T')) \wedge \\ &\quad FulfillTerminable(O^{a,T_s} X) \wedge \\ &\quad (T_e = T_s + 1) \wedge (T' < T_s) \end{aligned} \tag{A13}$$

The predicate '*FulfillTerminable*' is a boolean switch that allows for checking whether or not the obligation can be terminated upon fulfillment. This leave us to determine the conditions under which we have a violation of an achievement obligation. To this end we need a special event $deadline(O^{a,T_s} X)$ signaling the deadline after which a violation occurs if the achievement is not fulfilled by that time/event.

$$\begin{aligned} &Happens(violation(O^{a,T_s} X), T_v) \leftarrow \\ &\quad DHoldsAt(O^{a,T_s} X, T_e) \wedge \\ &\quad Happens(deadline(O^{a,T_s} X), T_e) \wedge \\ &\quad (\neg Happens(X, T_e) \wedge \neg HoldsAt(X, T_e)) \wedge \\ &\quad FulfillTerminable(O^{a,T_s} X) \wedge (T_v = T_e) \end{aligned} \tag{A14}$$

Maintenance Obligation Maintenance is another case of persistent obligation where it is different from achievement in the sense that the obligation conditions must be fulfilled for every instance of the interval the obligation is in force. The (Axiom A8) can represent the maintenance obligation. Contrary to achievement obligation, a maintenance obligation is violated if the obligation contents are not fulfilled for all the instances.

$$\begin{aligned} &Happens(violation(O^{m,T_s} X), T_k) \leftarrow \\ &\quad DHoldsAt(O^{m,T_s} X, T_k) \wedge \\ &\quad \neg Happens(X, T_k) \wedge \neg HoldsAt(X, T_k) \wedge (T_s \leq T_k) \end{aligned} \tag{A15}$$

The violation of a maintenance obligation may terminate the obligation if the obligation is '*ViolationTerminable*' which is again a boolean switch for checking whether a maintenance obligation can be terminated upon violation. The conditions for termination after the violation are:

$$\begin{aligned} &DTerminates(O^{m,T_s} X, T_v) \leftarrow \\ &\quad Happens(violation(O^{m,T_s} X), T_v) \wedge \\ &\quad ViolationTerminable(O^{m,T_s} X) \end{aligned} \tag{A16}$$

For a non-perdurant maintenance obligation the violation of the obligation itself terminates the obligation.

$$\begin{aligned} &DTerminates(violation(O^{m,T_v}X),T_v) \leftarrow \\ &\quad DHoldsAt(O^{m,T_v}X,t_v) \wedge ViolationTerminable(O^{m,T_s}X) \wedge \\ &\quad Happens(violation(O^{m,T_s}X),T_v) \wedge (T_s \leq T_v) \end{aligned} \tag{A17}$$

4.3 Compensation Obligation

A compensation is an obligation itself. The event triggering a compensation is the violation of a norm compensation compensates. Thus, we have domain specific axioms for the two case of compensation:

- *Compensation of the violation by a single obligation*:

$$\begin{aligned} &Happens(compensation(O^{x,T_s}P),T_{s_c}) \leftarrow \\ &\quad \exists O^{y,T_{s_c}}Q : (Compensates(O^{y,T_{s_c}}Q,O^{x,T_s}P),T_{s_c}) \wedge \\ &\quad Happens(violation(O^{x,T_s}P),T_v) \wedge \\ &\quad DHoldsAt(O^{y,T_{s_c}}Q,T_{s_c}) \wedge \\ &\quad (Happens(Q,T_{s_c}) \vee HoldsAt(Q,T_{s_c})) \wedge (T_s \leq T_v \leq T_{s_c}) \end{aligned} \tag{A18}$$

- *Recursive compensation when a compensation obligation itself is violated*:

$$\begin{aligned} &Happens(compensation(O^{x,T_s}P),T_{s_c}) \leftarrow \\ &\quad Compensates(O^{y,T_{s_c}}Q,O^{x,T_s}P) \wedge \\ &\quad Happens(violation(O^{y,T_{s_c}}Q),T_v) \wedge \\ &\quad Happens(compensation(O^{y,T_{s_c}}Q),T_z) \wedge \\ &\quad RecursivelyCompensable(O^{x,T_s}P) \wedge (T_s \leq T_{s_c} \leq T_z) \wedge (T_v \leq T_z) \end{aligned} \tag{A19}$$

For the two axioms above we have to introduce the special event *compensation*, indicating that a (violated) deontic fluent has been compensated for, and the binary predicate *Compensates* where the two arguments are two deontic fluents. The meaning of *Compensates* is that fulfilling the first deontic fluent make amend to the violation of the second deontic fluents and implements the *Comp* function introduced in Section 2, Definition 7. Again the predicate *RecursivelyCompensable* is a boolean switch meant to capture the intuition given by condition 2 of Definition 9.

5 Proof Sketches of Correctness

The aim of this section is to show how to prove the correctness of our formalisation of norms in EC and the classificatory conditions of Section 2. For space reasons we provide only the proof sketch of the axioms for punctual obligation. The proofs for the remaining axioms are essentially similar.

First we introduce some base conditions relating to the basic predicates of EC and the functions *Force* and *State* providing thus the basic bridge between the axiomatisation in Section 4 and the conditions in Section 2.

C1. $HoldsAt(X,T)$ if and only if $X \in State(T)$,
C2. $Happens(X,T)$ if and only if $X \in State(T)$,
C3. $DHoldsAt(X,T)$ if and only if $X \in Force(T)$.

Lemma 1 (Punctual Obligation). *If $DHoldsAt(O^{p,T_s}X,T_s)$ is true, then X is a punctual obligation in Force at time T_s, $X \in Force(T_s)$*

Proof (Sketch). By Definition 3 the semantics of a punctual obligation is given by (A) $o \in Force(n)$, (B) $o \notin Force(n-1)$, (C) $o \notin Force(n+1)$. Suppose, we have the right hand side of Axiom **??**, from this we obtain $DHoldsAt(O^{p,T_s}X,T_s)$, then from condition $C3$ we have $DHoldsAt(X,T)$ if and only if $X \in Force(T)$ which is equivalent to $X \in Force(T_s)$, and then $\exists n$ such that $X \in Force(n)$. This satisfies (A).

For (B), we assume that $\neg DHoldsAt(O^{p,T_s}X,0)$, where 0 is the initial time instant. By Axiom **??** this guarantees that the fluent $O^{p,T_s}X$ is not in *Force* function before the time, i.e., T_s. This means that $X \notin Force(t)$, for $0 \leq t < T_s$; hence $X \notin Force(T_s - 1)$. This satisfies condition (B).

Given that the right hand side of Axiom **??** is the same as that of Axiom **??**, we have $DTerminates(trigger(O^{p,T_s},N),T_s)$. From Axiom **??** we conclude $\neg DHoldsAt(O^{p,T_s},T_s+1)$. From condition C3 above we get $X \notin Force(T_s+1)$, which satisfies conditions (C).

Lemma 2 (Violation of Punctual Obligation). *If $Happens(violation(O^{p,T_s}X),T_s)$ is true, then X is a punctual obligation in force at time T_s, and $X \notin State(T_s)$.*

Proof (Sketch). To have a violation of a punctual obligation o, conditions (A), (B), (C) of Lemma **??** have to be satisfied and the additional condition (D) $o \notin State(n)$. That $Happens(violation(O^{p,T_s}X),T_s)$ is true means that also the right hand side of Axiom **??** is true. Thus we have $DHoldsAt(O^{p,T_s}X,T_s)$, from which we conclude the $X \in Force(T_s)$ by Lemma **??** above. In addition we have $\neg HoldsAt(X,T_s)$ and $\neg Happens(X,T_s)$ from which by conditions C1 and C2 above we conclude $X \notin State(T_s)$. This satisfies condition (D).

6 Related Work

In [6], EC is used to express temporal rules about the obligations and permissions in a business process interaction. Rich axioms that translate the temporal properties of deontic assignments and capture the effects of activities of obligations and permissions on the agents have been proposed. The study is limited in scope because it only covers obligations and permissions while other obligations types have been left out. Also, the temporal validity of an obligation and its effects on the violation, as presented in our work, has not been considered. Such parameters and ability to faithfully model obligations, and capture the effects of violations is imperative from a business process compliance checking perspective. [4] provides formal specifications of commitments and precommitments, instutionalised power and context using EC. The formal representation of norms is limited to obligations and permissions only as in [6]. No explicit distinction between the different types of obligations and effects of the violation on obligations has been made, as made in this work, although the notion of sanctions has been formally presented in the study.

[2] translates both the policies and system behaviour specifications into formal specifications using EC. The proposed formal specifications are expressive enough to

efficiently model the systems using various types of policies representing obligations. These formal specifications can be used, together with abductive reasoning, for detecting and representing the conflicts between the policy specifications (particularly those related to the authorisation and permissions). These specifications are useful in the sense that a priori knowledge about the event and/or fluent's state can be used to simplify the representation of preemptive obligations but we do not consider the a priori knowledge of events/fluents instead we use the notion of preemptiveness to distinguish different cases of the violation of an achievement obligation and model it in EC. [1] proposes a norms representation approach using EC enabling the agents to use norms in their practical reasoning. The work considers only two classes of norms: *obligations* and *prohibitions* for which authors introduced three fluents i.e., $fPun$ and $oPun$ referring obligation norm violation and prohibition norm violation respectively, and $oRew$ for obligation fulfillment. The scope of this work is limited because it only considers obligations and ignores the obligations modalities as we do. Also, the Anderson's reduction view of norm which suggests that every violation of a norm is followed by a sanction [20] has been used. We argue that initially not in every case sanctions are/can be directly imposed as under a sub-ideal situation processes can still be compliant [11]. The notions of compensation and obligations perduring after the violation as defined in our work are the norms types that strengthen this argument.

7 Final Remarks

In this paper we formally modeled the various types of obligations using classical EC. We used these obligations types from our previously proposed classification model, and introduced a triggering event (*trigger*) with some time delay replacing the *Initiates*, base predicate of the EC. The aim of the triggering event is to capture the deontic effects of obligations from when they enter into force not from when the event is triggered, which in our view is not possible with the existing variants of EC. The new predicates extend the expressive power of the EC and make it possible to model all types of legal norms. We are currently working on an implementation to validate the computational efficiency of the proposed extension to EC. Accordingly, we plan to continue this work and check the expressive power of various formalisms e.g., temporal logic, first-order-logic and defeasible and deontic logic. Also, we will look at the *state of affairs* in the formal modeling of the legal knowledge and what is lacking in this direction.

References

1. W. Alrawagfeh. Norm Representation and Reasoning: A Formalization in Event Calculus. In G. Boella, E. Elkind, B. T. R. Savarimuthu, F. Dignum, and M. Purvis, editors, *PRIMA 2013*, volume 8291 of *LNCS*, pages 5–20. Springer, 2013.
2. A. Bandara, E. Lupu, and A. Russo. Using Event Calculus to Formalise Policy Specification and Analysis. In *POLICY 2003*, pages 26–39, 2003.
3. DECLARE. Declarative Process Models, http://www.win.tue.nl/declare/.
4. N. Fornara and M. Colombetti. Specifying artificial institutions in the event calculus. In *Handbook of Research on Multi-Agent Systems: Sematnics and Dynamics of Organisational Models*, pages 335–366. IGI Global, 2009.

5. S. Goedertier and J. Vanthienen. Business Rules for Compliant Business Process Models. In *(BIS 2006)*, volume P-85 of *LNI*, pages 558–579. Gesellschaft für Informatik, 2006.
6. S. Goedertier and J. Vanthienen. Designing Compliant Business Processes with Obligations and Permissions. In J. Eder and S. Dustdar, editors, *Business Process Management Workshops 2006*, LNCS 4103, pages 5–14. Springer, 2006.
7. G. Governatori. Business Process Compliance: An Abstract Normative Framework. In *it-Information Technoloby*, volume 55(6), pages 231–238, 2013.
8. G. Governatori, J. Hulstijn, R. Riveret, and A. Rotolo. Characterising Deadlines in Temporal Modal Defeasible Logic. In *Proceedings of the 20th Australian Joint Conference on Advances in Artificial Intelligence*, AI'07, pages 486–496. Springer, 2007.
9. G. Governatori and Z. Milosevic. Dealing with Contract Violations: Formalism and Domain Specific Language. In *EDOC 2005*, pages 46–57. IEEE Computer Society, 2005.
10. G. Governatori, A. Rotolo, and G. Sartor. Temporalised Normative Positions in Defeasible Logic. In *ICAIL'05*, pages 25–34. ACM, 2005.
11. G. Governatori and S. Sadiq. The Journey to Business Process Compliance. In *Handbook of Research on Business Process Management*, pages 426–454. IGI Global, 2009.
12. M. Hashmi, G. Governatori, and M. T. Wynn. Normative Requirements for Business Process Compliance. In *ASSRI'13*, November 2013. [to appear].
13. M. Hilty, D. A. Basin, and A. Pretschner. On Obligations. In *ESORICS*, pages 98–117, 2005.
14. R. Kowalski and M. Sergot. A Logic-Based Calculus of Events. In J. Schmidt and C. Thanos, editors, *Foundations of Knowledge Base Management*, Topics in Information Systems, pages 23–55. Springer, 1989.
15. R. Miller and M. Shanahan. The Event Calculus in Classical Logic - Alternative Axiomatisations. *Electron. Trans. Artif. Intell.*, 3(A):77–105, 1999.
16. R. Miller and M. Shanahan. Some Alternative Formulations of the Event-Calculus. In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond*, volume 2408 of *LNCS*, pages 452–490. Springer, 2002.
17. M. Palmirani, G. Governatori, and G. Contissa. Modelling Temporal Legal Rules. In *ICAIL*, pages 131–135, 2011.
18. F. Sadri and R. Kowalski. Variants of the Event Calculus. In L. Sterling, editor, *Proceedings of the Twelth International Conference on Logic Programming*. MIT Cambrdige, 1995.
19. G. Sartor. *Legal Reasoning: A Cognitive Approach to the Law*. Springer, 2005.
20. A. Soeteman. Pluralism and Law. In *Proceedings of the 20th IVR World Congress of the Int'l Association of Philosophy of Law and Social Philosophy*, volume 4, page 104, 2001.