# A Methodology for Formalizing Different Types of Norms*

Soheil Roshankish(✉), Nicoletta Fornara

Università della Svizzera italiana, via G. Buffi 13, 6900 Lugano, Switzerland
soheil.roshankish@usi.ch,nicoletta.fornara@usi.ch

**Abstract.** In a world where many activities are carried out digitally, it is increasingly urgent to be able to formally represent the rules, norms, and policies that regulate these activities. In multi-agent systems, formalizing policies written in a natural language into a formal model, making them machine-readable, is a demanding task. In this paper, we introduce a methodology to help people to understand the fundamental elements that they should consider for this transformation. In this paper we will focus mainly on a methodology for formalizing norms using the T-Norm norm model, this because it allows us to express a rich set of different types of norms. In any case, the proposed methodology is general enough to also be used, in some of its steps, to formalize norms using other formal languages. This is an important issue because since there is not yet a set of different types of norms that is sufficiently expressive and is recognized as valid by the NorMAS community, papers presenting a given model usually do not explicitly state which types of norms can be expressed with that model and which cannot. Therefore, the second goal of this paper is to propose and discuss a rich set of norm types that could be used to study the expressive power of different formal models of norms and to compare them.

## 1 Introduction

In a world where many activities are carried out digitally, it is increasingly urgent to be able to formally represent the rules, norms, and policies that regulate these activities. By doing that, it is important to take into account that they are carried out by autonomous subjects who are able to decide to violate these rules. The activities regulated by the norms can be of various types and also include the very important ones related to the use, exchange and manipulation of the enormous amount of digital data that exist nowadays. Since these norms and policies can be violated (for example, it is very difficult to regiment obligations [8]) it is also urgent to propose mechanisms to automatically monitor compliance with these norms.

In the academic literature, there are an interesting number of general models of norms and policies. Some of these are close to being able to be used in real

* Funded by the SNSF (Swiss National Science Foundation) grant no. 200021_175759/1.

applications in today's Web as they are expressed with standard Semantic Web Technologies, which is a crucial characteristics for realizing interoperable systems. One of them is the W3C Recommendation ODRL 2.2[1] (the Open Digital Rights Language), which is a policy expression language that provides an information model and a vocabulary for specifying permissions, prohibitions, and obligations about the usage of digital assets and services. Two others are the T-Norm model [10] and the OWL-POLAR model [19]. They are two semantic web based complementary models having an operational semantics for reasoning about norms and policies fulfillments and violations.

The papers that describe these models are mainly focused on the presentation of the model that is exemplified usually with the formalization of some examples of norms, regardless of their type. What is missing, however, is a *methodology* that explains what steps should be followed if one wants to start from a norm written in a natural language (e.g., English) and be able to choose the model for its formalization and use it to arrive at the formal specification of the norm, which can then be used to reason about it and verify its fulfillment or violation. Since there is not a commonly accepted set of *types of norms* in the literature, papers presenting a given model usually do not explicitly state which types of norms can be expressed with that model and which cannot. Thus in this paper we have the following two goals.

Our first goal is to explain the methodology that can be used by people to translate norms written in one natural language into a language specifically designed for the formal specification of norms. The proposed methodology consists in: first understand if the norm can be expressed with a certain model, that means to understand which *type* the norm belongs to and if the type is supported by the model; secondly come to a proper formalization of the norm using the chosen model, this by applying the methodology proposed in this paper. We will focus mainly on formalizing norms using the T-Norm norm model, this because it allows us to express a rich set of different types of norms. However, it is important to emphasize that the proposed methodology is general enough to also be used, in some of its steps, to formalize norms using other formal languages that have some similarities with the T-Norm model, such as at least OWL-POLAR and ODRL.

There are many reasons why it is interesting to specify norms using formal models. First, because norms become machine-readable, therefore it is possible to automatically analyse and query them like for example it is discussed in [15] where the PrivOnto ontology is used for analysing 115 privacy policies. For example, it will be possible to search the set of resources on which it is possible to perform certain actions based on the customers' interests. When a policy is formalized with a machine-readable formal language that has an operational semantics, it is also possible to provide services for compliance checking of policies [19, 16, 9, 10]. This functionality plays an important role especially in domains in which the customers' sensitive data is collected and companies need to monitor the compliance of customers' privacy. This functionality is important also to cre-

[1] `https://www.w3.org/TR/odrl-model/`

ate a trustworthy environment for customers by providing monitoring platform that they can use to see whether their privacy policies (norms) are violated or not. For instance, a customer can attach to one picture the prohibition to publish it on a public platform for advertisement and would like to monitor if the actions which are performed on the picture are compliant with this prohibition.

Another reason why it can be useful to specify norms with formal languages is that it becomes easier for humans to understand their actual meaning, which is not always as immediate as it should be. For example, during the Covid-19 pandemic, it was not always easy to immediately understand what norms are in effect at any given time in a specific location and whether they entail obligations or prohibitions to perform actions. Another example of norms whose meaning and implications are not always immediately clear to the reader are the various privacy policies that regulate the processing of our data when we browse websites and use social networks. Users often accept such policies in order to use online services often without fully understanding what they mean, this is because they are too long or complex.

The second goal of this paper is to propose and discuss a rich set of norm types that could be used to study the expressive power of different formal models of norms and to compare them. Knowing that a certain model of norms is or is not capable of expressing a certain type of norms is fundamental to deciding which model to adopt in a certain application context. For example, if a norm generates a specific obligation that has a deadline, it is necessary to choose a model of norms that allows to express this temporal constraint and to verify its fulfillment. Secondly, once it is clear that a certain type of norm can be expressed in both language A and language B, it will also be possible to translate norms written with the first language into norms written with the second. Thus making systems that use different norm models interoperable, a fundamental aspect in today's world where one software agent must be able to interact with multiple open interaction systems without having to be reprogrammed every time.

This paper is organized as follows: in Section 2 the most relevant and recent papers presenting a model for norms and policies specification in which semantic web technologies have been used are discussed. In Section 3 the T-Norm model is briefly presented. In Section 4 the methodology proposed in this paper is described and the set of different types of norms is discussed. Finally in Section 5 we draw some conclusions.

## 2 Related works

In the multiagent systems community, over the last twenty years, many models of norms and policies for regulating the behaviour of autonomous agents have been proposed [2, 6]. In some of these models semantic web technologies have been used for modeling some components of norms/policies that can be used for expressing obligations, prohibitions, and permissions. The first proposals were the KAoS framework [21], the REI [14] policy language, and the PRovisional TrUst NEgotiation framework Protune [4]. Those approaches are summarized and compared in [5] where the requirements for a policy framework are discussed and the various approaches are categorized discussing whether the policies are public or not. For example, for the public policies, it is possible to use KAOS and REI frameworks as we need just one step evaluation to see if two policies are compatible. On the other hand, if a policy contains sensitive data, they are required to have *stateful and stateless negotiation* protocols for further security concerns.

An important policy language based on semantic web technologies, which is a W3C Recommendation since 15 February 2018, is the Open Digital Rights Language (ODRL 2.2). It is a policy expression language that can be used to represent permitted, prohibited, and obliged actions over a certain asset. ODRL policies may be limited by constraints (e.g., temporal or spatial constraints). ODRL was originally (in 2001) an XML language for expressing digital rights, that is, digital content usage terms and conditions. In version 2.0 and 2.1 ORDL is a Policy Language formalized in RDF with an abstract model specified by an ontology.It has no formal semantics, so compliance checking of policies written with this language cannot be performed automatically. An interesting attempt to give a formal semantics to ODRL 2.1 policies is presented in [20]. Some extensions of ODRL has been proposed to overcome to some of its limits. In particular, in [9] an extension of the ODRL Information Model has been proposed together with a set of state machines used for describing the evolution in time of the deontic state of obligations, prohibitions, and permissions. Another extension of ODRL is presented in [7] to model both regulatory policies (in the form of nested permissions, prohibitions, obligations and dispnesations), and business policies via discrete permissions. A policy written with that extension of the ODRL language is then translated into an Institutional Action Language (InstAL) policy and thanks to its formal semantics, expressed in Answer Set Programming, it is possible to automatically check compliance and also provide an explanation of the aspects of the policy that brings to the non-compliance. In [13] a specific use case drawn from the social networks field is used to validate the expressiveness of the ODRL 2.0 model.

Other two interesting proposals of a policy/norm model and framework, which are based on semantic web technologies, are OWL-POLAR [19] and T-Norm [10, 11] models. Those policies/norms models and their expressivity will be discussed in Section 3. An interesting aspect that differentiates the two models is the way in which the two models define mechanisms to reason about policies to test whether agents' behavior satisfies them or not. In the OWL-POLAR a query answering mechanism (DL-safe) has been used to check if any action that happened satisfies the policies. In the T-Norm model a rule-based approach is used that brings the generation of several deontic relations used to represent obligations and prohibitions generated by the activations of norms. In addition, the T-Norm model makes it possible to formalize the temporal constraints that exist between the activation of a norm and the class of actions regulated by the norm.

Other interesting models of norms that, like the T-Norm model, are rule-based are: the one proposed by Garcia-Camino et al. [12] where rules are operationalized using the JESS a rule engine for the Java platform; and the one proposed in [1] where reasoning on norms is realized with DROOLS a business rule management system. Another interesting proposal is the OASIS standard LegalRuleML[2], which defines a rule interchange language for the legal domain and is formalized using RuleML.

Logic and Knowledge Engineering Framework and Methodology (LOGIKEY) is another interesting framework which was introduced recently in [3]. The main objective of this framework is to enable and support the practical development of computational tools for normative reasoning based on formal methods. In their approach, they use higher-order logic (HOL) as the formal framework. They also used some GDPR examples to show how their framework supports the ethical and/or legal (ethico-legal) domains theories.

## 3 The T-Norm model

The *T-Norm* model can be used to formalize a precise and rich set of *types of norms* that regulate the interactions between autonomous agents. Namely (as we will further discuss in the paper): (i) norms with a activation condition represented by a class of events; (ii) norms without an activation condition; norms that generate (iii) general or (iv) specific obligations or prohibitions to perform (or not to perform) actions that can be constrained to happen before something else happens, (v) exceptions to those norms; (vi) exceptions to obligations and prohibitions (i.e. exemptions and permissions respectively). Once a set of norms is formalized, a specific *framework* has been proposed to automatically check if the agents' behavior conforms or does not conform to the given set of norms. This is done by simulating or monitoring the evolution of the state of those set of norms as time passes, events occur and autonomous agents perform actions. The framework for norms monitoring has been proposed by taking into account the *operational semantics* of the T-Norm model. The model, its operational semantics, and the framework were introduced in [10, 11].

The *T-Norm* model captures the following intuitive meaning of norms: whenever a particular *activation condition* is satisfied (i.e. an event that belongs to a particular class of events occurs) a *deontic relation* (general or specific) is created for regulating the performance of a class of actions by certain agents. In turn, every time an action belonging to the *class of the regulated actions* is executed before a certain event happens (for example a certain temporal event representing a deadline) and the deontic relationship represents an obligation it will be *fulfilled*, while if the deontic relation represents a prohibition it will be *violated*. On the contrary, when an action belonging to the class of regulated actions can no longer be performed (for example because the deadline has expired) and the deontic relationship represents an obligation it will be *violated*, while if the deontic relation represents a prohibition it will be *fulfilled*.

[2] `https://www.oasis-open.org/committees/legalruleml/`

In order to formally describe such a dynamic behaviour, the abstract model of a norm cannot consists only of a set of facts (like it is in many models of norms and policies, e.g. ODRL[3], OWL-POLAR [19], and the model proposed in [1]). In all these models the intrinsically dynamic nature of norms is described in their semantics or is left to their intuitive meaning described in the text. The T-Norm model allows to specify how the performance of certain actions or the occurrence of certain events will change the state of the interaction among agents. Therefore the basic building blocks of the T-Norm abstract norm are *rules* of the form `ON...THEN...ELSE`[4]. The abstract norm has not a pre-defined deontic type, as discussed in Section 4, it is those who formally specify a norm who will decide whether the norm activation creates obligations or prohibitions. In the T-Norm model a generic abstract norm has the following form:

```
----------------------------------------------------------------
Abstract Norm
----------------------------------------------------------------
 1:  NORM Norm_n
 2:  [ON ?event1 WHERE conditions on ?event1
 3:   THEN
 4:     COMPUTE]
 5:     CREATE DeonticRelation(?dr);
 6:     ASSERT isGenerated(?dr,Norm_n); [activated(?dr,?event1);]
 7:     ON ?event2 [BEFORE ?event3 WHERE conditions on ?event3]
 8:        WHERE actor(?event2,?agent) AND conditions on ?event2
 9:     THEN ASSERT fulfills(?agent,?dr); fullfilled(?dr,?event2)|
10:                 violates(?agent,?dr); violated(?dr,?event2)
11:  [ELSE ASSERT violates(?agent,?dr); violated(?dr,?event3)|
12:               fulfills(?agent,?dr); fulfilled(?dr,?event3)]
```

In the proposed model the first (optional) `ON...THEN` component (line 2,3) is used for expressing those norms that have an activation condition. The second `ON...THEN` component (line 7,9) is used for expressing that when a specific action, which belongs to the class of actions regulated by the norm, is performed (before something else occurs) there will be a fulfillment or a violation. In alternative, the `ELSE` part of the second rule (line 11) will be followed when an action that belongs to the class of the regulated actions can no longer be performed.

The *formulas* used in the abstract norm are conjunctions (in the `WHERE` part) or sequences (in the `CREATE` and `ASSERT` part) of *atomic assertions* written using the classes (unary predicates starting with capital letter) and the properties (binary predicates starting with a lowercase letter) defined in the *T-Norm Ontology* depicted in Figure 1[5]. Variables (starting with '?') refers to individuals. Variables used in the `WHERE` parts of the norm for expressing `conditions` on events

[3] https://www.w3.org/TR/odrl-model/

[4] The ON clause has been chosen instead of the more common IF clause to highlight that the part after THEN is executed when a particular event or action occurs and not simply when a condition is satisfied

[5] The T-Norm ontology in OWL is available at `https://raw.githubusercontent.com/fornaran/T-Norm-Model/main/tnorm.owl`

can be used freely and have to be bound to individuals in the *State Knowledge Base* (where the interaction among agents is represented) for the conditions to be met. In the WHERE parts it is also possible to compare the value of a variable with a constant value using any of the symbols $\{<, >, =, \neq, \leq, \geq\}$. A constant is a numerical value or an individual in the ontology. Variables that appear in the `ASSERT` part of a norm must have been introduced previously in one of its `ON` or `CREATE` parts. In the `COMPUTE` part some values can be calculated (for example the deadlines) using the value of previously introduced variables[6].

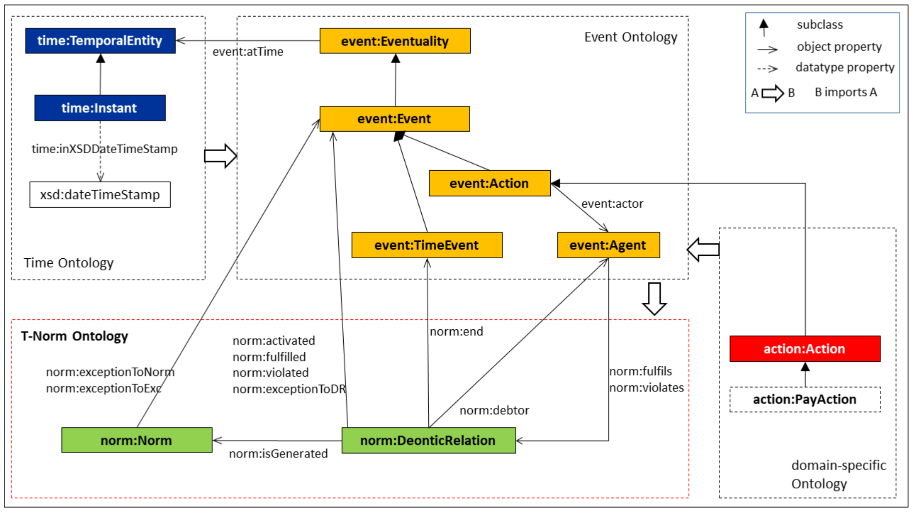

**Fig. 1.** The T-NORM Ontology and its connections with other ontologies.

## 4 Methodology

In this section, we describe the various steps of the procedure to be followed to transform a norm written in a natural language (for example in English) into a norm written using a formal machine-readable language like the *T-Norm* model. As it will be discussed, some steps of the described procedure can also be used to formalize norms using the OWL-POLAR or the ORDL normative language. Starting from a norm expressed in natural language, following each step of the

[6] The choice of using conjunctions or sequences of atomic assertions (analogously to what is proposed in OWL-POLAR to express the various components of their norm model) has the advantage of avoiding requiring the user of the model to learn a specific formal language, once written those expressions can be easily and automatically translated into the conditions or actions of production rules or into SPARQL queries.

methodology, the *Abstract Norm*, introduced in the previous section, is made more concrete to the point of being the formalization of the norm from which the process started. In the following, we use two real running examples. We call the first example Norm1, which is inspired from the law regarding the access to limited traffic area in Milan city such that "*when an agent enters in the limited-traffic area of Milan, between 7:30 and 19:30, they have to pay 5 euros before 24:00 on the day of entry*"[7]. The second example, called Norm2, is the rule that must be followed by libraries in Italy regarding lending of DVDs, it is "*Italian libraries cannot lend DVDs until 2 years are passed from the distribution of the DVD*"[8].

## 4.1 Using ontologies for modeling norms

In the first step of the procedure, we need to define (or search among the existing ones) one or more formal ontologies to represent the classes of events or actions that are relevant for the norm that we want to formalize. As it is discussed in the previous section, for every norm normally three *classes of events/actions* should be specified:

- The class of events that represent the activation condition of the norm (described using `?event1` in line 2 in the *Abstract Norm*);
- Te class of actions regulated by the norm (described using `?event2` in line 7,8 in the *Abstract Norm*);
- The class of events defined for constraining the performance of the actions regulated by the norm. One action, belonging to the class of the regulated ones, should or should not occur before an event belonging to this constraining class (described using `?event3` in line 7 in the *Abstract Norm*).

Those classes are specified in the `WHERE` parts of the norm and are represented using the classes and properties defined in formal ontologies. In the *T-Norm* model and in *OWL-POLAR* model, the W3C Web Ontology Language (OWL 2) is used for specifying ontologies. OWL is a Semantic Web language designed to represent knowledge about things, groups of things, and relations between things. An important advantage of using OWL is that it is a well-known standard language, which can make it easier for those who want to formalize their norms. Moreover, the formal semantics of OWL makes it possible to perform automatic reasoning on the state of the world, an operation that has important consequences on the computation of the *deontic force* (it is obliged or it is prohibited) associated to the actions performed by the agents. In ODRL the information model of the language is formalized using an OWL ontology, while the actions, the parties, and the assets involved in one ODRL policy are described using the ODRL Vocabulary[9].

[7] https://www.comune.milano.it/aree-tematiche/mobilita/area-c
[8] According to Art. 69 c.1 of the Copyright Law (22-4-1941, no. 633)
[9] `https://www.w3.org/TR/odrl-vocab/`

It is possible to use several different ontologies for representing class of actions and their properties inside one T-Norm norm or one OWL-POLAR policy. For compatibility reasons, we suggest to use ontologies that are compatible with OWL ontologies, this because the chosen ontologies (each one referred to as *domain-specific Ontology* in Figure 1) should be imported into the *Event Ontology* which is written by using the OWL language.

We will now exemplify the formalization of two classes of events necessary for the formalization of `Norm1`. `Norm1` is activated every time a vehicle enters the restricted traffic zone of the city of Milan. We assume that `RestrictedTraffic AreaAccess` is a class of actions, `vehicle` and `owner` are two properties having as domain the `RestrictedTrafficAreaAccess` class, which are defined in an OWL domain-specific Ontology. We can then specify the class of events that activates `Norm1` as (where `?event1` is shortened to `?e1`):

```
ON ?e1 WHERE RestrictedTrafficAreaAccess(?e1) AND vehicle(?e1,?v) AND
owner(?v,?agent) AND atTime(?e1,?inst1) AND inXSDDateTimeStamp(?inst1,?t1)
AND ?t1.time()>07:30:00 AND ?t1.time()<19:30:00
```

In the previous expression the variable `?agent` is introduced because it will be used in the second part of the norm to recognize who fulfills or violates the norm. After representing the activation condition of the norm, we need to formalize the class of actions regulated by the norm. For `Norm1`, the class of actions is the payment of 5 euros before 24:00 on the day of entry. For formalizing it we can for example use the `PayAction` class defined in the *Schema.org* vocabulary, which has an OWL version. *Schema.org* provides a collection of types and properties available at the URL `schema.org`. The major search engine providers use the Schema.org markup to improve the searching and the display of search results. This vocabulary has been designed by and is controlled by these organizations and represents an interesting attempt to realize a lightweight ontology that can be reused in different applications.

As mentioned earlier, the class of events described with the variable `?event3` has the role of constraining the time interval in which the action belonging to the class of actions regulated by the norm (`?event2`) shall or cannot be performed. In `Norm1`, the time interval when the payment action should be performed is constrained by a deadline (referred with the variable `?paymentDeadline`), i.e. the payment action must occur before 24:00 on the day of entry into the limited traffic zone. The formalization of norms where `?event3` is a time event are discussed in Section 4.4. To be able to express facts about the topological relationships (ordering) between instants and intervals, along with information about their length, and their value in terms of dates and times, we imported the *W3C Time Ontology*[10] into our *Event Ontology* both written in OWL. However, in the formalization of other norms the time constraint could be expressed with any class of actions (e.g. the payment must be made before leaving the restricted area) and in this case we will need to use another ontology to represent that class. The

[10] https://www.w3.org/TR/owl-time/

class of actions regulated by `Norm1` can be specified as (where `?e1` and `?agent` are variables introduced in the previous `ON` clause):

```
ON ?e2 BEFORE ?paymentDeadline
   WHERE PayAction(?e2) AND reason(?e2,?e1) AND recipient(?e2,Milan)
   AND price(?e2,5) AND priceCurrency(?e2,euro) AND actor(?e2,?agent)
```

### 4.2 Norms with activation condition

The goal of this section is to explain how to recognize whether a norm has an *activation condition* or not. In every norms or policy models the activation condition may describe a *class of events/actions* or as a *state of affairs*.

In the T-Norm model, the activation condition is the description of a particular class of events or actions. When an event that belongs to the activation condition class occurs, a new deontic relation is created and some temporal parameters may be computed. In order to recognize the activation condition in the text of a norm, we have to look for the events or actions that induce the model to activate obligations or prohibitions. The temporal relation between an event or action that satisfy the activation condition and the action that should or should not be performed is crucial: the activation condition must be satisfied before the obligation or the prohibition to perform a certain class of actions starts to hold. The instant of time at which the activation condition of a norm is satisfied by an event or action is very important because it can be used to calculate the deadline of obligations generated by the norm or the instant of time at which a prohibition ceases to subsist. For example in Norm1, the activation condition is represented by the class of actions regarding entering the Milan limited traffic zone and it is used for computing the deadline for the payment. In Norm2, the activation condition is given by the class of actions with which a DVD distribution is initiated. It is important to note that in the T-Norm model the activation condition cannot describe *a state of affairs*, although often a state of affairs is the result of an event and therefore the description of the class of events may substitute the description of the state of affairs as activation condition.

The reason why, in a norm model, a class of events and a state of affairs are treated differently is mainly due to their ontological difference: an event when it has happened it can no longer be retracted, a state of affairs can be satisfied at a certain instant of time and it can become unsatisfied subsequently. This is a crucial difference, because in the T-Norm model any satisfaction of the activation condition leads to the permanent creation of deontic relations. This permanent creation is important when the deontic relation regulates a class of actions that should or should not be performed in an interval of time and when the deontic relation itself can generate many violations and fulfillments as it is discussed below when Norm2 will be formalized.

The OWL-POLAR model for policies can be used when the activation condition $\alpha$ describes a state of affairs, like for example "a person is obliged to leave a location when there is a fire risk" or "when a person has a child which is under 18 they have to pay their tuition"[19]. In the OWL-POLAR model a policy is activated for a specific agent when the world state is such that the activation condition holds for that agent and the expiration condition does not hold. Therefore, at the time of activation it is necessary to know the specific agent for whom the policy is being activated, and this is not the case when the activation of a policy leads to the creation of *general deontic relations*, as will be discussed below. In the OWL-POLAR model the time constraint that exists between the initial satisfaction of the activation and the subsequent activation of the policy is not explicitly represented in the norms model, it is expressed in the description of how it is possible to reason about policies, therefore it is not in the model but inside the algorithm proposed for reasoning on policies.

There is another important distinction between an activation condition that describes a class of events and the one that describes a state of affairs. When it describe a state of affairs, it may make sense to ask whether the condition in the text should be formalized as an activation condition or as a set of conditions that restrict the class of actions regulated by the policy. For example in [18] *conditional norms* are discussed and the following example of a norm with a condition is described as: "it is prohibited to litter as long as there is a rubbish bin within x meters from an agent". The condition of being within x meters from a rubbish bin may be modelled as an activation condition in some cases, and it can be considered as a condition that constrains the class of actions regulated by the policy in other cases. In this second case the policy can be modelled with the T-Norm model, and it is a prohibition (without activation condition) to perform the following class of actions: littering when the actor of the action is within x metres from the rubbish bin. If one action belonging to that class is performed then there is a violation of the prohibition.

The choice between the first and the second formalization depends on the type of reasoning that the norm designer[11] wants to be able to perform on the policy. In the first case (with an activation condition) it is possible to compute if the policy is active in a given situation and therefore plan the action for fulfilling or violating it. When computing all the activations may be too costly and the goal of reasoning is monitoring the fulfillment or violation of the policy, the second formalization, without activation condition, is the more efficient because it does not require to compute the activation of many policies.

In the ODRL 2.2 model it is possible to express *constraints* associated to the rules contained in one policy and *refinements* associated to the actions regulated by one rule (a duty, a prohibition, or a permission). Reading the documentation that provides the meaning of a prohibition or duty, the constraint can be used to express the activation condition, but again, like in OWL-POLAR, the temporal constrains between the satisfaction of the activation condition and the performance of the action regulated by the rule is not explicitly expressed in the model. Therefore, when the activation condition is a state of affairs the policy designer has to choose weather it is better to put the conditions in the constraint or in the refinement. Differently, when the activation condition of the norm is represented

[11] The term "norm designer" refers to the person in charge of formalizing norms with a formal model.

by a class of events, by using the ODRL 2.2 and OWL-POLAR models it is impossible to specify in the norm formalization the need to compute at run-time the value of the deadline and it is impossible to model those policies that when are activated generates *general* deontic relations.

## 4.3 Representing obligations and prohibitions

In this section, we describe how we can distinguish if a norm generates an obligation or prohibition and how to express them using the T-Norm model. In contrary with other approaches such as OWL-POLAR and ODRL, in the T-Norm model there is not a component or a predefined class that may be used to specify if the norm express an obligation or a prohibition. The advantage of this approach is that both obligation and prohibitions can be expressed starting from the same abstract norm and there is not need to formalize the semantics or a state machine (like in [9]) for obligations, another one for prohibitions, and others for other deontic concepts like permission, right, privilege, liability and so on. However, in this Section we focus only on obligation and prohibition.

In the T-Norm model the intuitive meaning of having an obligation or prohibition is that when something happens and certain conditions hold, an agent is obliged or prohibited to do something in a given interval of time. We can use few basic constructs and combine them in different ways to express the obligation to perform an action before a given deadline or the prohibition to perform an action within an interval of time. The main difference in representing a prohibition or an obligation is in the second `THEN` part of the norm. If the norm designer wants to formalize the obligation to perform an action, performing the regulated action must bring to the specification of the fulfillment of the deontic relation in the `THEN` part of the norm. The `ELSE` part have to be used to specify that in case on instance of the class of actions regulated by the policy cannot be performed before than a given event happens the deontic relation, representing the obligation, becomes violated. On the contrary, if the norm designer wants to formalize the prohibition to perform an action (described in the second ON part of the norm) in the specific interval of time, performing the action will bring to the violation of the deontic relation in the second `THEN` part. Once the prohibited action can no longer be performed (for example, the time interval has expired) the prohibition becomes fulfilled.

As we know from deontic logic literature[22] the expression “it is impermissible (IM) that *p*” is defined as equivalent to “it is obligatory (OB) that not p” ($IMp = defOB\neg p$). This implies that some norms may be either formalized as an obligation or as a prohibition. When a norm is formalized with the T-Norm model and the activation of the norm brings to the creation of *general deontic relations*, it is very important to evaluate which of the two formalizations would be most cost-effective. That is because, as discussed below, every general deontic relation created by the activation of a norm, may in turn bring to the costly generation of many fulfillments and violations. For example, the norm “when the school bell rings, students should go back to the classrooms in five minutes” can be formalized as a norm that generates obligations or prohibitions.

Suppose that the person in charge of formalizing the norm is only interested in computing the violations of the norm. In the first scenario, if we formalize the norm as a generator of obligations, when the activation condition is satisfied because the school bell rings, the norm generates a general deontic relation that will generate fulfillments for all those students who respect the school rule and go back to their classrooms, and violations for those students who did not fulfill the rule before the deadline. In the second scenario, it is possible to formalize such a norm as a generator of prohibitions by reframing it as follows "when five minutes have elapsed since the bell rang, students cannot remain in the courtyard". The formalization of this norm is much easier and cost-effective as we only need to check the violations that are generated for those students who stay in the courtyard.

### 4.4 Temporal aspect of norms

The ability to represent time-constrained norms is one important characteristics of the T-Norm model. Unlike the OWL-POLAR model, in the T-Norm model can be used by the norm designer to easily represent any obligations containing deadlines (that are represented as time events) and prohibitions that holds for an interval of time. A norm governs a class of actions and, as can be seen from the abstract norm in Section 3, that class can be temporally constrained by specifying the `BEFORE` part and another class of events (`?event3`). The latter class can be specified using the TimeEvent class or the more generic Event class depicted in Figure 1.

The TimeEvent class is used for specifying a deadline for obligations or the instant at when a time interval for prohibitions ends. In this case the value of the deadline or the end of the time interval can be computed in the `COMPUTE` part of the norm as exemplified below. For example, in `Norm01`, an agent is obliged to perform the *paying* action before midnight (the deadline). In `Norm02`, the time interval in which Italian libraries cannot lend DVDs begins with the release of the DVD and ends after 2 years. Norm2 can be represented with the T-Norm model as follows:

```
----------------------------------------------------------------------
Norm2
----------------------------------------------------------------------
ON  ?e1
 WHERE isReleased(?e1) AND object(?e1,?dvd) AND VideoObject(?dvd) AND
 place(?e1;Italy) AND atTime(?e1,?inst1) AND inXSDDateTimeStamp(?inst1,?t1)
THEN
 COMPUTE ?tend_n=?t1.year+2
 CREATE DeonticRelation(?dr);TimeEvent(?tev_end_n);Instant(?inst_end_n);
 ASSERT isGenerated(?dr,Norm2); activated(?dr,?e1); end(?dr,?tevend_n);
        atTime(?tev_end_n,?inst_end_n);
        inXSDDateTimeStamp(?inst_end_n,?tend_n);
 ON ?e2 BEFORE ?tev_end_n
  WHERE LendAction(?e2) AND object(?e2,?dvd) AND actor(?e2,?agent)
 THEN  violates(?agent;?dr); violated(?dr,?e2)
```

The `CREATE` and `ASSERT` parts of the norm above, which specify the characteristics of the time event used to constrain the class of actions governed by the norm, represent a prototype of what these two parts look like in all such type of norms.

On the other hand, if the temporal constraint (?event3) belongs to a generic `Event` class (or one of its subclasses), it is not necessary to compute anything. This means that the regulated action is temporally constrained by another generic class of events. For example, in the norm “You should pay the parking ticket before exiting”, there exists no deadline for the payment action, but the payment action must be performed before leaving the parking area with one’s car. This event should be specified in the WHERE part used to describe `?event3`.

In literature there exist approaches, such as [17], in which they used temporal logics such as Linear Temporal Logic (LTL) for representing time-constrained norms. Nevertheless, using these approaches present some difficulties when it comes to the automatic reasoning on the evolution of the normative state from activated to fulfilled or violated.

## 4.5 Specific and General Deontic Relations

In the T-Norms model, a norm can create several deontic relations when the activation condition of the norm is satisfied. Such deontic relations may belong to one of the following two categories: *specific deontic relations* and *general deontic relations*.

A specific deontic relation is generated, when the regulated action should be performed by a specific agent, e.g. in `Norm1`, for each vehicle entering into the limited traffic area an obligation to pay for the owner of the vehicle is generated. In the specific deontic relations, the debtor, the owner of the vehicle, is known. Therefore, in case of any violation, the system can recognize who violated the deontic relation. The specific deontic relations generated by `Norm1` have a `debtor` property that connects the deontic relation with the agent that is the owner of the vehicle: `ASSERT` ... `debtor(?dr, ?agent)`.... This property has to be inserted in the `ASSERT` part of the norm for all those norms that generate specific deontic relations.

The second type of deontic relations is the general deontic relations. The main difference between general deontic relations and specific deontic relations is that in the first one we do not have any knowledge about the debtor of the class of actions regulated by the norm and the action can be performed by a set of agents, for example in Norm2 by all the people registered in one library. For that reason, we cannot have any predefined estimation about which agent is going to violate or fulfill the deontic relation. It is possible to have many violations and many fulfillments of the same deontic relation. For example `Norm2` is activated for every new distribution of a DVD. The general deontic relation created by the activation of `Norm2` regulates the actions of lending such a DVD by all the Italian libraries, the debtor is not one specific agent.

Another significant difference between OWL-POLAR and the T-Norm model is in formalizing norms that generates general deontic relations. In OWL-POLAR a policy can only be activated for a specific agent therefore the type of norms that when activated regulate the actions of a set of agents cannot be represented. This is due to the design choice to propose a model for reasoning on policies that does not create deontic relations. In ODRL it not specified the mechanism for reasoning on policies activations.

## 5 Conclusions

In this paper, we introduced a methodology that explains how a norm designer can formalize norms written in a natural language into a machine-readable format by understanding the types of the norms and choosing the proper model. As it is discussed in the previous section the norm can be: (i) a norm with or without an activation condition; (ii) if there is an activation condition it can be represented by a class of events or by a state of affairs; (iii) a norm can express obligations or a prohibitions; (iv) a norm can regulate a class of actions that is time constrained or not; (v) finally a norm can generate specific or general deontic relations. In our future works we plan to extend the methodology by discussing the formalization of exceptions to norms and in particular of permissions and exemptions and the definition of institutional powers for manipulating norms.

## 6 Acknowledgement

The Version of Record of this contribution is published in Multi-Agent Systems vol 13442. Springer, Cham, and is available at `https://doi.org/10.1007/978-3-031-20614-6_20` .

## References

1. S. Álvarez-Napagao, H. Aldewereld, J. Vázquez-Salceda, and F. Dignum. Normative monitoring: Semantics and implementation. In M. D. Vos et al., editors, *COIN@AAMAS 2010, Toronto, Canada, May 2010, COIN@MALLOW 2010, Lyon, France, August 2010, Revised Selected Papers*, volume 6541 of *LNCS*, pages 321–336. Springer, 2010.
2. G. Andrighetto, G. Governatori, P. Noriega, and L. W. N. van der Torre, editors. *Normative Multi-Agent Systems*, volume 4 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013.
3. C. Benzmüller, X. Parent, and L. W. N. van der Torre. Designing normative theories for ethical and legal reasoning: Logikey framework, methodology, and tool support. *Artificial Intelligence*, 287:103348, 2020.
4. P. Bonatti and D. Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. In *Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)*, pages 14–23, 2005.

5. P. A. Bonatti and D. Olmedilla. *Rule-Based Policy Representation and Reasoning for the Semantic Web*, pages 240–268. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
6. A. Chopra, L. van der Torre, H. Verhagen, and S. Villata, editors. *Handbook of Normative Multiagent Systems*. College Publications, Aug. 2018.
7. M. De Vos, S. Kirrane, J. Padget, and K. Satoh. ODRL Policy Modelling and Compliance Checking. In P. Fodor, M. Montali, D. Calvanese, and D. Roman, editors, *Rules and Reasoning*, pages 36–51, Cham, 2019. Springer International Publishing.
8. N. Fornara and M. Colombetti. Specifying and enforcing norms in artificial institutions. In M. Baldoni, T. C. Son, M. B. van Riemsdijk, and M. Winikoff, editors, *Declarative Agent Languages and Technologies VI*, pages 1–17, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
9. N. Fornara and M. Colombetti. Using semantic web technologies and production rules for reasoning on obligations, permissions, and prohibitions. *AI Commun.*, 32(4):319–334, 2019.
10. N. Fornara, S. Roshankish, and M. Colombetti. A framework for automatic monitoring of norms that regulate time constrained actions. In *Proceedings of the International Workshop on Coordination, Organizations, Institutions, Norms and Ethics for Governance of Multi-Agent Systems (COINE), co-located with AAMAS 2021, 3rd May 2021, London, UK, 2021*. arXiv, 2021.
11. N. Fornara and M. Sterpetti. An architecture for monitoring norms that combines OWL reasoning and forward chaining over rules. In E. M. S. et al., editor, *Proceedings of the Joint Ontology Workshops 2021 Episode VII: The Bolzano Summer of Knowledge co-located with the 12th International Conference on Formal Ontology in Information Systems (FOIS 2021), and the 12th International Conference on Biomedical Ontologies (ICBO 2021), Bolzano, Italy, September 11-18, 2021*, volume 2969 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021.
12. A. Garcia-Camino, P. Noriega, and J. A. Rodriguez-Aguilar. Implementing norms in electronic institutions. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '05, pages 667–673, New York, NY, USA, 2005. ACM.
13. G. Governatori and R. Iannella. A modelling and reasoning framework for social networks policies. *Enterp. Inf. Syst.*, 5(1):145–167, feb 2011.
14. L. Kagal. *A Policy-Based Approach to Governing Autonomous Behavior in Distributed Environments*. PhD thesis, University of Maryland Baltimore County, Baltimore MD 21250, September 2004.
15. A. Oltramari, D. Piraviperumal, F. Schaub, S. Wilson, S. Cherivirala, T. B. Norton, N. C. Russell, P. Story, J. R. Reidenberg, and N. M. Sadeh. Privonto: A semantic framework for the analysis of privacy policies. *Semantic Web*, 9(2):185–203, 2018.
16. J. Padget, M. D. Vos, and C. A. Page. Deontic sensors. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 475–481. International Joint Conferences on Artificial Intelligence Organization, 7 2018.
17. S. Panagiotidi, S. Alvarez-Napagao, and J. Vázquez-Salceda. Towards the norm-aware agent: Bridging the gap between deontic specifications and practical mechanisms for norm monitoring and norm-aware planning. In *Revised Selected Papers of the COIN 2013*, volume 8386, pages 346–363. Springer-Verlag Inc., 2014.
18. B. T. R. Savarimuthu, S. Cranefield, M. Purvis, and M. K. Purvis. Identifying conditional norms in multi-agent societies. In M. D. Vos, N. Fornara, J. V. Pitt,

and G. A. Vouros, editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems VI - COIN 2010 International Workshops, COIN@AAMAS 2010, Toronto, Canada, May 2010, COIN@MALLOW 2010, Lyon, France, August 2010, Revised Selected Papers*, volume 6541 of *Lecture Notes in Computer Science*, pages 285–302. Springer, 2010.

19. M. Sensoy, T. J. Norman, W. W. Vasconcelos, and K. P. Sycara. OWL-POLAR: A framework for semantic policy representation and reasoning. *Journal of Web Semantics*, 12:148–160, 2012.
20. S. Steyskal and A. Polleres. Towards Formal Semantics for ODRL Policies. In N. Bassiliades, G. Gottlob, F. Sadri, A. Paschke, and D. Roman, editors, *RuleML 2015, Berlin, Germany, August 2-5, 2015, Proceedings*, volume 9202 of *Lecture Notes in Computer Science*, pages 360–375. Springer, 2015.
21. A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott. Kaos policy and domain services: toward a description-logic approach to policy representation, deconfliction, and enforcement. In *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pages 93–96, 2003.
22. G. H. von Wright. Deontic logic. *Mind, New Series*, 60(237):1–15, 1951.