Hindawi
Complexity
Volume 2019, Article ID 7895875, 20 pages
https://doi.org/10.1155/2019/7895875

Research Article

A Norm Compliance Approach for Open and Goal-Directed Intelligent Systems

Patrizia Ribino and Carmelo Lodato

Istituto di Calcolo e Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche, Via Ugo La Malfa, 153, 90146 Palermo, Italy

Correspondence should be addressed to Patrizia Ribino; patrizia.ribino@icar.cnr.it

Received 3 December 2018; Accepted 11 February 2019; Published 8 April 2019

Academic Editor: Xianming Zhang

The increasing development of autonomous intelligent systems, such as smart vehicles, smart homes, and social robots, poses new challenges to face. Among them, ensuring that such systems behave lawfully is one of the crucial topics to be addressed for improving their employment in real contexts of daily life. In this work, we present an approach for norm compliance in the context of open and goal-directed intelligent systems working in dynamic normative environments where goals, services, and norms may change. Such an approach complements a goal-directed system modifying its goals and the way to achieve them for taking norms into accounts, thus influencing the practical reasoning process that goal-oriented systems implement for figuring out what to do. The conformity to norms is established at the goal level rather than at the action level. The effect of a norm that acts at the goal level spreads out at the lower level of actions, thus also improving system flexibility. Recovery mechanisms are provided to face exceptional situations that could be caused by normative changes. A case study in the field of the business organizations is presented for demonstrating the strengths of the proposed solution.

1. Introduction

Intelligent systems are increasingly used with some degree of autonomy. Thus, their behaviour is not entirely defined by the designer, but it is the result of cognitive capabilities. Modern intelligent systems can determine by themselves the behaviour to adopt for achieving their objectives. In doing so, they could be involved in behaviours that in real-world life are regulated by more or less stringent norms that could produce different effects. For example, smart cars have to obey the city traffic laws. Intelligent information systems have to comply with data protection laws to manage private information. Social robots have to respect social norms during interaction with humans. Smart workflow management systems have to respect business rules to perform business processes and so on.

A growing issue in the field of artificial intelligence is how to ensure that the behaviour of intelligent systems complies with the normative environment they should operate so that these systems can be well-accepted and efficiently employed in real contexts of the everyday life [1].

In conventional approaches, norms are fully specified at design-time, and the system is designed in such a way that its behaviour does not violate such rules. These approaches, based on hard-coded static norms, are not practical solutions because all the possible situations that a system has to manage should be established at design-time. Otherwise, system redesigning is necessary [2]. Most advanced approaches use model-checking techniques for verifying the system behaviour off-line. Such methods result in impracticable or limited when the autonomy of the system increases making its behaviours not entirely predicted. Moreover, because normative environments in which systems operate are increasingly dynamic, to avoid the shutdown of the system and its reconfiguration, norm compliance has to be guaranteed at run-time.

In this paper, we present an approach for ensuring system run-time compliance with a dynamic set of norms in the context of open and goal-directed systems. The conformity with norms is guaranteed at a higher level of abstraction (i.e., the goal level). Such an approach complements a goal-directed system modifying its goals and the way to achieve

them for taking norms into account, thus influencing the practical reasoning process [3] that the goal-oriented systems implement for figuring out what to do. In our approach, we also faced the problem to manage normative changes during system execution that may produce system incoherence with existing norms providing recovery mechanisms. In this work, we refined some theoretical foundations that have been preliminarily presented in [4], and we introduced new ones for defining the algorithms that implement the proposed approach. A widely known case study in the context of business organizations [5] is also presented for illustrating the behaviour of a goal-oriented system that implements our algorithms.

The rest of the paper is organized as follows. Sections 2 and 3 present some grounding literature and the theoretical background of the paper, respectively. In Section 4, a general overview of the approach is presented. Sections 5 and 6 present the key concepts and the algorithms for normative reasoning. Section 7 illustrates a case study in the context of business organizations. Finally, in Section 8 conclusions are drawn.

2. Related Works

Norms like obligations, permissions, and prohibitions have been implemented in automatic systems to specify (un)desired or (un)lawful behaviours. Normative systems [6] are commonly defined as systems that specify every possible transition, whether or not that transition is considered to be legal or not. They determine which actions or which states should be achieved or avoided [7–9].

Much work has been done about normative frameworks in the field of Electronic Institutions or Virtual Organizations where norms have found a natural implementation. Only to cite a few, Alechina et al. [10] present a programming framework for developing normative organizations based on N-2APL, a BDI-based agent programming language supporting normative concepts such as obligations, prohibitions, and sanctions. In such a work, the interaction between agents and the environment is regulated by a "normative exogenous organization" which is defined using a set of conditional norms. A norm-aware deliberation approach is also proposed. It allows agents to determine the set of plans (adopted for satisfying a goal) of highest priority which does not violate higher priority prohibitions. In [11], Kollingbaum and Norman proposed the Normative Agent Architecture (NoA). It supports the implementation of norm-governed practical reasoning agents. NoA agents are motivated by norms to act. In the NoA language, all the effects of a plan are declared in a plan specification. These effects are considered by agents for reasoning about plan selection and execution. Moreover, the norms governing the behaviour of an NoA agent refer to actions that are obligatory, permitted, forbidden, or states of affairs that are obligatory, permitted, or forbidden. The NoA language enables an agent to be programmed in terms of plans and norms. Normative statements formulated in the NoA language express obligations, permissions, and prohibitions of an agent. In [12], the problem of regulating the operation of open multiagent systems in which multiple interrelated activities take place is addressed, thus involving the distributed management of norms. Authors propose normative structures as a means to observe and manage the evolution of normative positions in each activity and their propagation in distributed activities. In [13], authors treated a computer supported cooperative work (CSCW) system as an organization in a society to use the abstraction of an organizational model. Hence, they propose a logical predication-based organizational model with an organizational state machine (OSM) to describe norms in CSCW systems. The organizational model allows describing the behavioural rules of roles, and the OSM allows for checking the logical conflict among the rules.

Some other works have been conducted for addressing norm change and norm consistency providing agents with mechanisms for enacting behaviour modification. Typically, new plans/actions have been created to comply with new norms [14–16]. In [17], Jiang et al. propose a normative structure, named *Norm Nets* (NNs) for modeling sets of interrelated regulations. NNs are aimed at verifying whether executions of business processes complied with process regulations. Authors define a norm as a tuple of elements that specify the type of deontic operator, the pair role-action (the target) to which the deontic modality is assigned, a deadline of norm validity, and a precondition that determines when the target is initiated. A formal method for checking norm compliance by using Colored Petri Nets is proposed. In [18, 19], authors propose a means for automatically detecting and solving conflict and inconsistency in norm-regulated Virtual Organization and Electronic Institution.

This work is developed in the context of the open and goal-directed systems that are able to work in dynamic normative environments and to organize their behaviour according to environmental changes.

The approach we propose is founded on a norm compliance algorithm, starting from the knowledge about norms, goals, and state of the world and acting at the goal level, which allows the system to plan the right behaviour in conformity with the normative context. Such an algorithm also allows the system to address exceptional situations that may occur, mainly when several norms act simultaneously. Three possible cases we considered in detail: inconsistent norms that contain a logical contradiction, the presence of an antinomy, namely, a conflict between two norms that are mutually exclusive, and, finally, norms that are incompatible with system requirements, in other words, that make goals not satisfiable.

3. Background

This section is organized in two parts. The first one introduces norms and normative reasoning. The second one gives some details about antinomies in the legal theory.

3.1. Normative Reasoning. Norms are everywhere in our daily life. We are obligated to follow the traffic regulations during driving. We have to respect contractual restrictions at work.

We try to follow etiquette to be well-accepted in society, and so on. These examples spread from hard to soft constraints. Generally speaking, norms can be seen as normative rules used for governing conducts, procedures, or state of affairs within a particular sphere of knowledge. The normative reasoning is the reasoning conforming to or based on norms. It is formally studied through the deontic logic [20], which is the study of the logical relationships among propositions asserting that specific actions or state of affairs are obligatory, forbidden, or permitted. From the application point of view, deontic logic is the logic that deals with actual as well as the ideal behaviour of systems [21]. Differently from the classic logic, when working with norms the main issue to be considered is that norms are neither true nor false. For overcoming this issue, a common approach is to consider the deontic logic as a logic of normative propositions. The underlying concept is that, even if norms are neither true nor false, someone may state that something ought to be done (according to norms): the statement *Mary ought to pay the taxes* is, then, true or false description of a normative situation [22].

Standard deontic logic [23, 24] is the most studied system of deontic logic and one of the first deontic logics axiomatically specified. It is obtained from the modal logic and employs three modal operators O, P, and F, where OA means that *it is obligatory that A*, PA means that *it is permitted that A* and FA that *it is forbidden that A*. Let O be primitive, the operators P and F can be defined by the equivalences $PA \equiv \neg O \neg A$ and $FA \equiv O \neg A$, where PA means that *it is not obligatory that not A* and FA means that *it is obligatory that not A*. In this context, A designates a proposition that asserts that an act of the sort A is done. Thus, OA is read as "it is obligatory that the situation described by the descriptive sentence A is realized".

3.2. Antinomy in Legal Theory. In legal theory, an antinomy is defined as an incompatibility relation between two norms belonging to the same legal system. If we refer to the classic logic, the incompatibility between two propositions is determined by the impossibility that they are both true. In legal theory, the concept of incompatibility is founded on the deontic logic, which refers to the obligatory of the prescriptive assertions.

The deontic square of oppositions [24] shows all the possible relations between two norms in terms of obligations, prohibitions, permissions, and negative permissions (see Figure 1). The most common types of oppositions between the four modalities are as follows: *(1) Incompatibility between a norm that prescribes P and a norm that prohibits P*. Only one of them may be in effect at any time.

(2) Incompatibility between a norm that prescribes P and a norm that allows not P. An action is either obligatory or not. Omissible corresponds to the absence of an obligation (may $\neg(X) \equiv \neg$ must(X)) and vice versa.

(3) Incompatibility between a norm that prohibits P and a norm that allows P. An action is either forbidden or allowed. Permission therefore corresponds to the absence of a prohibition (may(X)$\equiv \neg$ (must *not*(X))). For solving antinomy,

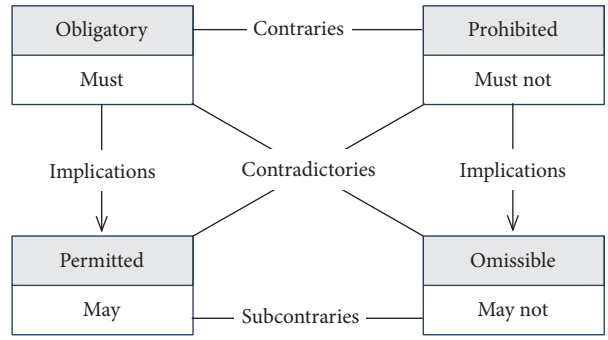

Figure 1: The deontic square of oppositions.

three criteria are adopted in legal theory. The *legis posterior* that states the younger law overrides the older law. The *legis specialis* establishes that a law governing a specific subject matter (*lex specialis*) overrides a law which only governs general matters (*lex generalis*). Finally, the *lex superior derogat legi inferiori* principle states that higher law overrides the lower law, because a legal system is commonly based on a power hierarchy. The proposed approach takes inspiration from this theory for addressing conflicting situations in the context of open and goal-directed intelligent systems.

4. Overview of the Proposed Approach

As previously said, the proposed work is developed in the context of open and goal-directed intelligent systems that work in dynamic normative environments. In this kind of systems, goals can be seen as motivators that provide them with the reason for doing something. Moreover, the open systems we considered may evolve at run-time because *(i)* new services could be made available for satisfying existing goals and *(ii)* new goals may be required to the system.

Hence, working in dynamic normative environments with systems that may evolve their behaviour requires new methods able to ensure norm compliance also for norms and system behaviours that are not defined at design-time. For addressing such issues, our approach is based on a triplet of elements, namely, state of the world, goal, and norm. A *state of the world* represents the particular conditions of the system and the context in which it works in a specific time. A *Goal* expresses the desired state of the world the system wants to achieve when certain conditions are verified. Finally, *Norms* regulate the state of the world using obligations, permissions, and prohibitions. In particular, we considered two different cases. In the first case, obligation and prohibition norms modify the desired state of the world according to the admissible state expressed by obligations or prohibitions. Permission may or may not change the desired state of the world. In the second case, norms are considered as promoters or inhibitors of the system in pursuing goals. Practically, a permission relaxes the constraints expressed by the conditions under which a goal has to be satisfied. Conversely, a prohibition nullifies the commitment with a goal under the circumstances defined by the prohibition norm. Finally, an

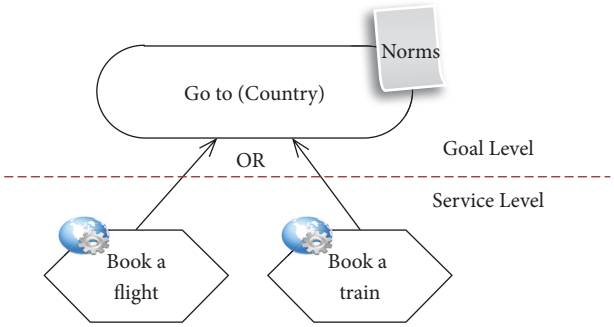

Figure 2: Services that could be chosen for satisfying the goal.

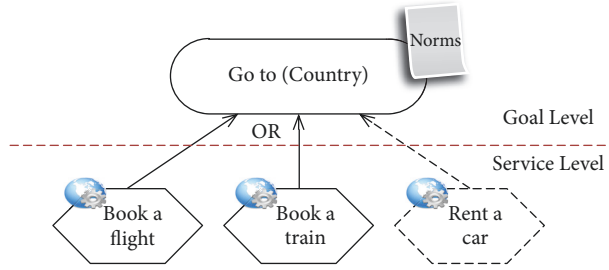

Figure 3: The availability of the new service *Rent a car* gives the system a new mean for satisfying the goal *Go to (country)*.

obligation introduces further conditions under which a goal has to be reached. In this vision, the effect of the norms is that they may increase the possibility for the system to pursue a goal (permissions). Conversely, they may inhibit system intentions to pursue a goal (prohibitions). Finally, norms may force the system to pursue a goal (obligations). The main feature of the proposed approach is to obtain a norm compliance behaviour by exploiting normative reasoning applied to the state of affairs a system wants to achieve. Thus, norms regulate the system at goal level providing some advantages and overcoming some limitations of conventional approaches.

For illustrating typical problems, we provide some examples (some norms used in the following examples are inspired to existing real norms). Let us suppose an intelligent system able to plan and organize a personal agenda of a user. Let us suppose that such a system can satisfy the user goal G_1 : *I want to go to country A*. We also assume the user has already obtained the entry visa for the country A that is a prerequisite for achieving G_1. Let us also suppose that, at the moment of the goal commitment, the system knows only two ways for satisfying the goal G_1: by booking a flight or a train. Figure 2 shows a goal model with services the system may use for producing a postsituation which satisfies G_1. We have also depicted two levels of abstractions to empathize the difference between goals and services.

Let us suppose that a norm N_1 states that "*It is prohibited that a person visits the country_A if he has visited a country_B*". Let us assume that the user has already visited the *country_B*. Thus, the presence of such norm does not permit the user to go to *country_A*. For complying with such a normative requirement, the system does not have to plan the user trip. The proposed approach allows the system to revise its current commitment to that goal. Hence, rather than disabling all the possible ways the system can follow to satisfy the goal, the norm applied at the goal level does not allow the system to pursue the goal because it inhibits its intentions.

Let us assume, instead, that the personal agenda is committed to fulfilling two user goals, *go to country_A* and *go to country_B*. For complying with $N1$, the personal agenda has to be able to plan the trip for country_A firstly and then the trip for country_B. Our approach allows the system to reason about the effect of the norm on its desired state of the world. Because choosing firstly to pursue the goal *go to country_B* leads the system in an unlawful state of the world, the system will plan the two goals opportunely.

Let us suppose now that a new service "*rent a car*" is available at run-time for satisfying the goal *"go to (country)"* (see Figure 3). The run-time introduction of this new element does not involve any change in the system configuration.

We do not need to modify anything to adapt the behaviour of the system to manage this new situation because the norm defined at the goal level spreads to the service level. The approach we propose allows maintaining norm compliance although the service level is changed.

Conversely, let us suppose that a norm N_2 states that *"In country A, it is prohibited to enter foreign cars"*. Let us suppose that available rent car services do not have cars produced in the *country A*. This kind of norm does not prohibit to pursue the goal G_1, but it has effects on it. In our approach, such kind of norm introduces constraints to the final state of the world the system wants to reach. Thus, the system will choose the *book a flight* or *book a train* service in order to achieve G_1.

Finally, let us suppose that a norm N_3 "*It is permitted that a person goes to country_A if he is a citizen of a member state of the organization X*" is at run-time introduced in the system. We also assume that the user has not an entry visa for the country A, yet. The single effect of N_3 on the system (i.e., without considering the presence of N_1 and N_2) is to relax the conditions under which the goal has to be satisfied. The system could plan to fulfill the goal *go to country_A* also without its prerequisite which is satisfied (i.e., without an entry visa for Country A). On the contrary, the simultaneous presence of N_1 and N_3 creates a joint effect on the system because they affect the same goal *"go to country_A"*. In particular, the injection of N_3 could cause a system deadlock. Indeed, if the conditions of N_1 and N_3 are simultaneously valid for the user, an antinomy is generated, and the system does not know how to behave. The compliance with N_3 causes to be uncompliant with N_1. It is a classic example of a conflict generated when two norms are contradictory. Our normative reasoning approach implements recovery criteria for addressing such situations. In particular, in this situation N_3 is a norm defined by superior institutional power. Thus it should prevail on the second one.

Such examples show only some of the situations that an open system should manage working in a dynamic normative environment. Several other anomalous situations determining system deadlock can occur in implementing norm

compliance. In this work, we address the following ones: (1) The first is *The introduction into the system of an inconsistent norm*. It means that the norm is self-contradictory because it contains a logical contradiction, namely, the conjunction of a statement S and its denied, not S. (2) The second is *The presence of an antinomy*. An antinomy designates a conflict of two norms that are mutually exclusive or that oppose one another. (3) The third is *The run-time injection of norms incompatible with a system goal*. This means that pursuing that goal always violates the prescribed norms.

In our approach, we manage these situations in dynamically changing environments, where conflicting situations among norms may change according to the particular execution context. For addressing this concern, we introduce some new definitions about conflicts and inconsistencies that are based on a representation of the execution context. The norm compliance approach is defined through algorithms that are based on these definitions. In the following section, the theoretical foundations of the approach are introduced.

5. Theoretical Foundations

This section formally introduces the theoretical foundations of the proposed approach. Firstly, the definitions of the state of the world, goals, and norms are introduced. Then, formal definitions about norm compliance and anomalous situations are presented. It is worth noting that such definitions imply a formalization of a *sphere of knowledge* within the system operates. In particular, to establish a relationship between the formal specification of goals and state of the world and the formal representation of the normative framework it is necessary to refer to the same semantic layer. This requirement can be satisfied by adopting a knowledge formalization based on proper domain ontology (it is out of the scope of the paper how to build the ontology. The reader may refer to the several approaches proposed in the literature).

5.1. State of the World, Goals and Norms. The *state of the world* represents a set of declarative information about events occurring within the environment and relations among events at a specific time. An event can be defined as the occurrence of some fact that can be perceived by or be communicated to the intelligent system. Events can be used to represent any information that can characterize the situation of an interacting user as well as a set of circumstances in which the intelligent system operates at a specific time.

Definition 1 (state of the world). Let $\mathscr{D}$ be the set of concepts defining a domain. Let $\mathscr{L}$ be a first-order logic defined on $\mathscr{D}$ with $\top$ a tautology and $\perp$ a logical contradiction, where an atomic formula $p(t_1, t_2 \ldots, t_n) \in \mathscr{L}$ is represented by a predicate applied to a tuple of terms $(t_1, t_2 \ldots, t_n) \in \mathscr{D}$ and the predicate is a property of or relation between such terms that can be true or false. A *state of the world* in a given time t ($\mathscr{W}^t$) is a subset of atomic formulae whose values are true at the time t:

$$\mathscr{W}^t = \left[p_1\left(t_1, t_2, \ldots, t_h\right), \ldots, p_n\left(t_1, t_2, \ldots, t_m\right)\right] \quad (1)$$

Definition 1 is based on the close world hypothesis [25] that assumes all facts that are not in the state of the world are considered false. Resuming the previous example, a possible state of the world at a given time t could be represented as

$$\mathtt{W}^{\mathtt{t}} = \left[\mathtt{has}\left(\mathtt{entry\_visa}, \mathtt{country\_A}\right), \mathtt{booked}\left(\mathtt{flight}, \mathtt{AR302}\right)\right] \quad (2)$$

It means that, at a given time t, the user has an entry visa for country_A and has received the confirmation of the flight $AR302$ reservation.

Definition 2 (goal). Let $\mathscr{D}$, $\mathscr{L}$, and $p(t_1, t_2 \ldots, t_n) \in \mathscr{L}$ be as previously introduced in Definition 1. Let $t_c \in \mathscr{L}$ and $f_s \in \mathscr{L}$ be formulae that may be composed of atomic formulae by means of logic connectives AND($\wedge$), OR ($\vee$), and NOT ($\neg$). A *Goal* is a pair $\langle t_c, f_s \rangle$ where t_c (*trigger condition*) is a condition to evaluate over a state of the world $\mathscr{W}^t$ when the goal may be actively pursued and f_s (*final state*) is a condition to evaluate over a state of the world $W^{t+\Delta t}$ when it is eventually addressed:

(i) *a goal is active if* $t_c(\mathscr{W}^t) \wedge \neg f_s(\mathscr{W}^t) = true$

(ii) *a goal is addressed if* $f_s(\mathscr{W}^{t+\Delta t}) = true$

A *Goal* describes a desired state of affairs the actor wants to achieve. It is represented by a couple of elements named trigger condition and final state. The final state represents the desired state of affairs. A goal is activated when some conditions occur (i.e., the trigger conditions) and it is satisfied when a new state of the world contains the final state.

The previous goal $G_1 = go\_to(country\_A)$ (according to common goal-oriented methodologies, we give a name to a goal for referring to it [26]) could be represented as follows:

$$G_1 = \left\langle \mathtt{has}\left(\mathtt{entry\_visa}, \mathtt{country\_A}\right), \mathtt{visited}\left(\mathtt{country\_A}\right)\right\rangle \quad (3)$$

It means that g will be actively pursued after receiving the *entry_visa* for *country A*. The goal is considered addressed in a new state of the world where the user is in *country A*.

Definition 3 (norm). Let $\mathscr{D}$, $\mathscr{L}$, and $p(t_1, t_2 \ldots, t_n) \in \mathscr{L}$ be as previously introduced in Definition 1. Let $\phi \in \mathscr{L}$ and $\rho \in \mathscr{L}$ be formulae composed of atomic formula by means of logic connectives AND($\wedge$), OR ($\vee$), and NOT ($\neg$). Moreover, let $D_{op} = \{permission, obligation, prohibition\}$ be the set of deontic operators. A *Norm* is defined by the elements of the following tuple:

$$n = \langle r, g, \rho, \phi, d \rangle_{[scope]}, \quad (4)$$

where *scope* identifies the field of reference of the norm; $r \in \mathscr{R}$ is the *Role* the norm refers to; $g \in \mathscr{G}$ is the *Goal* the norm refers to; $\rho \in \mathscr{L}$ is a formula expressing the set of actions and/or state of affairs that the norm disciplines; $\phi \in \mathscr{L}$ is a logic condition (to evaluate over a state of the world $\mathscr{W}^t$) under which the norm is applicable; $d \in D_{op}$ is the deontic

operator applied to ρ that the norm prescribes to the couple $(r, g) \in \mathscr{R} \times \mathscr{G}$:

$$d(\rho) = \begin{cases} \rho & \text{if } d = \textit{obligation} \\ \neg\rho, & \text{if } d = \textit{prohibition} \\ \rho \vee \neg\rho & \text{if } d = \textit{permission} \end{cases} \tag{5}$$

An obligation imposes to obtain the state of affairs ρ. A prohibition defines ρ as a not-acceptable state of affairs. Finally, permissions do not have a restrictive role. We also introduce the concept of scope for relating a norm to a specific context. The scope also allows establishing some hierarchy among norms.

The previous norm stating that "*It is prohibited that a person visits the country_A if he has visited a country_B*" could be represented as follows:

$$\begin{aligned} \mathtt{n}_1 = \langle &\mathtt{tourist}, \mathtt{g}_1, \mathtt{visited(country\_A)}, \\ &\mathtt{visited(country\_B)}, \mathtt{prohibition}\rangle_{[\mathtt{safety}]} \end{aligned} \tag{6}$$

Hereafter, we assume that given a norm $n = \langle r, g, \rho, \phi, d\rangle$; the actor that is pursuing the goal g plays the role r.

Definition 4 (applicable norm). Let $\mathscr{W}^t$ be a state of the world in a given time t. A norm $n = \langle r, g, \rho, \phi, d\rangle$ is *applicable* at time t if

$$\left(\phi\left(\mathscr{W}^t\right) = \textit{true}\right) \vee (\phi = \top) \tag{7}$$

Definition 4 establishes when in a given context a norm is workable. When the statement representing the applicability condition results in a tautology, the norm is applicable in each state of the world the system is.

Definition 5 (active norm). Let $\mathscr{W}^t$ be a state of the world in a given time t and let $g = \langle t_c, f_s\rangle$ be a not addressed goal. A norm $n = \langle r, g, \rho, \phi, d\rangle$ is *active* at time t if

$$n \textit{ is applicable at time } t \wedge \left(t_c\left(\mathscr{W}^t\right) = \textit{true}\right) \tag{8}$$

Definition 5 establishes when a given norm may produce an effect on the system. In particular, it occurs when the norm is applicable and the goal is active (namely, the system intends to pursue that goal). For the previous example, n_1 is applicable for a user that has already visited country B, but it may produce an effect only if the user has an *entry_visa* for *country A* that triggers the system to try to pursue the goal.

5.2. Norm Compliance. The definition of normative compliance is based on the concept of inadmissible state of the world defined as follows.

Definition 6 (inadmissible state of the world). A state of the world at a given time t

$$\mathscr{W}^t = \left[p_1(t_1, t_2, \ldots, t_h), \ldots, p_n(t_1, t_2, \ldots, t_m)\right] \tag{9}$$

is an *Inadmissible State of the World* if $\exists\, n = \langle r, g, \rho, \phi, d\rangle \mid Cond_A \wedge Cond_B$ are verified where

$$\begin{aligned} &Cond_A : \phi\left(\mathscr{W}^{t-\Delta t}\right) \wedge \neg\rho\left(\mathscr{W}^{t-\Delta t}\right) = \textit{true} \\ &\qquad\qquad\qquad\qquad \text{when } d = \textit{prohibition} \\ &Cond_B : p_1(t_1, t_2, \ldots, t_h) \wedge \cdots \wedge p_n(t_1, t_2, \ldots, t_m) \\ &\quad \wedge d(\rho) = \perp \end{aligned} \tag{10}$$

The first condition allows ensuring the nonretroactive effect of a prohibition norm. It disciplines the case where the state of affair regulated by a prohibition norm has occurred before the applicability of the norm. Therefore, if n is a prohibition and $\rho = \textit{true}$ before the norm came into force, the state of the world cannot be considered inadmissible. The second condition verifies if $\mathscr{W}^t$ is contrasting with the deontic constraint the norm prescribes.

It is worth noting that because of ρ may refer to a state of affairs then it might coincide with the desired state of the world (i.e., $\rho = f_s$). This means that the norm disciplines directly to the goal fulfillment. Conversely, when $\rho \neq f_s$, the norm constrains the way to reach the final state of the world by pursuing the goal. In the following definition, we differentiated these two cases.

Definition 7 (norm compliance). Let us consider a norm $n = \langle r, g, \rho, \phi, d\rangle$ and a goal $g = \langle t_c, f_s\rangle$. Let us consider a state of the world $\mathscr{W}^t$ in a given time t in which n is active and let $\mathscr{W}^{t+\Delta t}$ be the state of the world in which f_s is true. Pursuing the goal g is compliant with the norm n if $\mathscr{W}$ is an admissible state of the world where

$$\mathscr{W} \equiv \begin{cases} \mathscr{W}^{t+\Delta t} & f_s = \rho \\ \bigcup_{k=t}^{t+\Delta t} \mathscr{W}^k & f_s \neq \rho \end{cases} \tag{11}$$

The first case allows ensuring that the final state of the world achieved by pursuing a goal does not contain any violations of the normative constraints. The second case allows establishing that the system moved along a path which satisfies the norm passing through various states of the worlds appropriately. Definition 7 is strictly correlated to practical reasoning of goal-oriented systems. *Practical reasoning* is reasoning directed towards actions; it is the process of figuring out what to do [3]. It consists of two activities: *deliberation*, deciding what goals to achieve, and *means-ends reasoning*, determining how to meet these goals. The central aspect of goal deliberation is "*How can the system deliberate on its goals to decide which ones shall be pursued?*" [27]. A goal-oriented system sees some of its goals merely as possible options. Goal deliberation has the task to decide which goals a system actively pursues, which ones it delays, and which ones it abandons. Conversely, means-ends reasoning are aimed at providing operationalization of goals. It is the process of deciding how to achieve a goal using the available means (e.g., actions, services, and resources). In our approach, a mean describes a particular trajectory in

terms of the state of the world the system may intentionally use to address a given result. Thus the system knows its effect on the state of the world. The definition we introduce about norm compliance directly influences the process of goal deliberation. The first condition of Definition 7 has a direct impact on the choice of goals that can be pursued. A system can deliberate to pursue a goal based on run-time conditions by envisaging the normative effects of the goal. Conversely, means-ends reasoning is a process that allows choosing the appropriate ways to fulfill a deliberated goal. The second condition of norm compliance is implicitly related to this process. A system can determine the way to reach a goal by envisaging the normative effects of available means that it can choose.

5.3. Anomalous Situations. This section formalizes the types of exceptional situations that can occur working with normative propositions.

Definition 8 (inconsistent norm). A norm $n = \langle r, g, \rho, \phi, d\rangle_{[scope]}$ is *inconsistent* if $\phi = \bot$.

A norm $n = \langle r, g, \rho, \phi, d\rangle_{[scope]}$ is inconsistent when it contains a logical contradiction. This means that its applicability condition ϕ contains the conjunction of a statement S and its denied not S. For example, let us suppose a norm N: *It is prohibited that a person enters in building site if he is unauthorized, and if he is without protection and he is authorized*. This norm contains the conjunction of two contradictory statements (i.e., he is an unauthorized user and he is an authorized user). In this case, the norm will always be not applicable without effect on the system behaviour. Such a situation could occur during the definition of norms with a complex condition, or a writing error could determine it. In the previous example, the corrected norm could be as follows: *It is prohibited that a person enters in building site if he is unauthorized or if he is without protection and he is authorized*.

Definition 9 (incompatibility). Let $g = \langle t_c, f_s\rangle$ be a not addressed goal. A norm $n = \langle r, g, \rho, \phi, d\rangle$ is *incompatible with* g if

$$\begin{aligned} &(i)\ \ \phi = \top \\ &(ii)\ \ f_s \wedge d(\rho) = \bot \end{aligned} \tag{12}$$

An incompatibility exists between a norm and a goal when pursuing the goal always violates the prescribed norm.

Let us consider a norm $n = \langle r, g, \rho, \phi, d\rangle$ and a goal $g = \langle t_c, f_s\rangle$; the following incompatibility cases may arise.

Case A. $f_s = \rho$, $d = Prohibition$ and $\phi = \top$; then n modifies the final state of g as follows:

$$f_s' = f_s \wedge \neg f_s \tag{13}$$

In this case, n prohibits the state of affairs f_s directly; as a consequence it forbids to pursue the goal g in any way. In some cases, this type of norm could be used for inhibiting some system behaviours, but a run-time injection of a norm whose applicability condition is erroneously written in such a way to determine a tautology may cause an undesired system deadlock. For example, let us consider the following norm *It is prohibited to go to country A*. It directly constrains the achievement of the goal *go to country A*.

Case B. $(f_s \neq \rho) \wedge (f_s \cap \rho \neq \varnothing)$, $d = Prohibition$ and $\phi = \top$; then n modifies the final state of g as follows:

$$f_s' = f_s \wedge \neg\rho \tag{14}$$

In this case, n indirectly constraints the achievement of the goal because n forbids to be in a state that is necessary for the accomplishment of the goal. For example, let us consider a simple example to remain in the context of the previous case. The norm *It is prohibited to be in motion* highlights the impossibility to pursue the goal *go to country A* in compliance with the norm.

Definition 10 (deontic contradiction). Let W^t be a state of the world at time t and let $n_1 = \langle r_1, g_1, \rho_1, \phi_1, d_1\rangle$ and $n_2 = \langle r_2, g_2, \rho_2, \phi_2, d_2\rangle$ be two norms where $r_1 = r_2$, $g_1 = g_2$, $\rho_1 = \rho_2$. Norms n_1 and n_2 are *deontically contradictory* if

$$\begin{aligned} &(i)\ \ \phi_1\left(\mathscr{W}^t\right) \wedge \phi_2\left(\mathscr{W}^t\right) = true \\ &(ii)\ \ d_1 \neq d_2 \end{aligned} \tag{15}$$

As previously said, an antinomy designates a conflict of two norms that are mutually exclusive or that oppose one another. Some norms can generate an antinomy under certain circumstances. In autonomous systems, such cases are not predetermined. Thus, systems have to be able to evaluate each particular situation at run-time. In the following, we discuss some possible scenarios that could occur during system execution that are related to the possible kinds of antinomy presented in Section 3.2.

Let n_1 and n_2 be two norms, $n_1 = \langle r_1, g_1, \rho_1, \phi_1, d_1\rangle$ and $n_2 = \langle r_2, g_2, \rho_2, \phi_2, d_2\rangle$, where $r_1 = r_2$, $g_1 = g_2 = g$, and $g = \langle t_c, f_s\rangle$:

Case C. If $d_1 = Prohibition$, $d_2 = Obligation$ and $\rho_1 = \rho_2 = \rho$, the joint effect of n_1 and n_2 modifies g as follows:

$$g' = \begin{cases} (t_c, f_s \wedge \neg\rho \wedge \rho) & if\ \phi_1 \wedge \phi_2 \\ (t_c, f_s \wedge \neg\rho) & if\ \phi_1 \wedge \neg\phi_2 \\ (t_c, f_s \wedge \rho) & if\ \neg\phi_1 \wedge \phi_2 \end{cases} \tag{16}$$

In the first case, the system is in a conflict situation because of the contemporaneous applicability of n_1 and n_2. Thus, there is no way to be compliant with both norms; the system adopts the recovery criteria. In the other cases, the final state is constrained to be compliant with the applicable norm.

Case D. If $d_1 = Obligation$, $d_2 = Permission$, $\rho_1 = \rho$, and $\rho_2 = \neg\rho$, the joint effect of n_1 and n_2 modifies g as follows:

$$g' = \begin{cases} (t_c, (f_s \wedge \rho) \vee (f_s \wedge \neg\rho \wedge \rho)) & if\ \phi_1 \wedge \phi_2 \\ (t_c, f_s \wedge \rho) & if\ \phi_1 \wedge \neg\phi_2 \\ (t_c, (f_s \wedge \neg\rho) \vee (f_s \wedge \rho)) & if\ \neg\phi_1 \wedge \phi_2 \end{cases} \quad (17)$$

In the first case, the system avoids a conflict situation pursuing g in such a way that the final state of the world includes the state of affair expressed by ρ. In other cases, n_1 and n_2 do not create a conflict.

Case E. If $d_1 = Prohibition$, $d_2 = Permission$, and $\rho_1 = \rho_2 = \rho$, the joint effect of n_1 and n_2 modifies the final state of g as follows:

$$g' = \begin{cases} (t_c, (f_s \wedge \neg\rho \wedge \rho) \vee (f_s \wedge \neg\rho)) & if\ \phi_1 \wedge \phi_2 \\ (t_c, f_s \wedge \neg\rho) & if\ \phi_1 \wedge \neg\phi_2 \\ (t_c, (f_s \wedge \rho) \vee (f_s \wedge \neg\rho)) & if\ \neg\phi_1 \wedge \phi_2 \end{cases} \quad (18)$$

In the first case, to be compliant with norms the system fulfills g in such a way to avoid the state of affair expressed by ρ. In the other cases, n_1 and n_2 do not create a conflict.

In particular, when $\phi_1 = true$ and $\phi_2 = true$ and $f_s = \rho$ the previous cases can be represented as follows.

Case F. If $d_1 = Prohibition$, $d_2 = Obligation$, $f_s = \rho_1$, and $\rho_1 = \rho_2$, the joint effect of n_1 and n_2 leads the final state of g in a contradictory state of the world:

$$f_s' = \neg f_s \wedge f_s \quad (19)$$

In this case, there is no way to be compliant with both norms; the system adopts the recovery criteria.

Case G. If $d_1 = Obligation$, $d_2 = Permission$, $f_s = \rho_1$, and $\rho_2 = \neg\rho_1$, the joint effect of n_1 and n_2 modifies the final state of g as follows:

$$f_s' = f_s \wedge \neg f_s \vee f_s \quad (20)$$

This case does not have a real effect on the system. In any case, the system pursues g.

Case H. If $d_1 = Prohibition$, $d_2 = Permission$, $f_s = \rho_1$, and $\rho_1 = \rho_2$, the joint effect of n_1 and n_2 modifies the final state of g as follows:

$$f_s' = f_s \wedge \neg f_s \quad (21)$$

Also, in this case, there is no way to be compliant with both norms. Indeed, the system should not pursue g for not violating any norms.

It is worth noting that a norm is *logically contradictory* when the contradiction concerns the logical conditions ($\phi \in \mathscr{L}$) under which the norm is applicable. On the contrary, we talk about *deontically contradictory* when the contradiction concerns the semantic meaning of the deontic operator ($d \in D_{op}$) the norms apply to.

6. Algorithms for Norm Compliance

In this section, the algorithms that implement norm compliance for open and goal-directed systems are presented. For space concerns, they are placed at the end of the paper.

Algorithm 1 implements the normative compliance. It ensures that the system behaves in conformity with the normative environment it is operating. The triple of elements it works is a *state of the world* $\mathscr{W}^t$, a *set of goal* $\mathscr{G}$ the system has to satisfy and finally a *set of norms* $\mathscr{N}$ the system has to obey to comply with the normative environment. Both $\mathscr{N}$, $\mathscr{W}^t$, and $\mathscr{G}$ may change during system execution. The state of the world may change due to some events that can occur or some actions that can be performed in the environment. The set of norms may change due to the introduction of new normative requirements that can come into force or due to the deletion of existing ones. The updating of the existing norms is regarded as new norms. For the scope of the paper, in the algorithm, we highlight the case of the introduction of new norms that is the most interesting case that can generate anomalous situations. Finally, the set of goals may change for satisfying new user requirements.

Step ① makes a preliminary check on new norms (if any) that are dynamically introduced in the system to avoid the presence of anomalies according to Definitions 8 and 9 (see Algorithm 2). Such step ensures that norms are consistent and there is no incompatibility with the goal they refer. Step ② is the core of the normative reasoning. The system is in the state of the world ($\mathscr{W}^t$). The norms may have effects on the system goals only if goals are not addressed yet. Thus, the set of applicable norms is filtered (i.e., $\phi(W^t) = true$). For each goal g_i the system has to satisfy, the set of associated norms ($\mathscr{N}_i$) is initially processed to separate norms where $f_s = \rho$ by norms with $f_s \neq \rho$ because they produce a different effect on the system behaviour. As previously said, when $f_s \neq \rho$ norms act as constraints on the final state of the world, in particular, the compliance has to be ensured for obligations and prohibitions. It is worth noting that when $f_s \neq \rho$, permissions are not taken into consideration. Permissions do not play a direct role in norm compliance because they cannot be violated. Conversely, when $f_s = \rho$, norms act as constraints on system goal fulfillment. In particular, norms are promoters or inhibitors of the system in pursuing goals. In this case, permissions, obligations, and prohibitions have to be considered. As previously said, prohibitions do not allow the system to pursue a goal; obligations further constrain the conditions under which a goal can be pursued, and permissions conversely relax such conditions. Then, three different situations can occur.

The most simple one (step Ⓐ) is that there are no norms for g_i. In such a case, there are no restrictions and the system can pursue the goal g_i. The second situation (step Ⓑ) is a basic case in which the set of norms contains only one norm. In such a case, there are two possible occurrences:

(i) Step ⓘ: $f_s = \rho$ then if the norm is a permission/obligation and it is applicable, the system can fulfill that goal even if the original $t_c(\mathscr{W}^t) = false$; if the norm is a prohibition, it further constraints the

```
Data: W^t, G, N
while system is running do
  N^t_injected ⟵ inject_new_norms
  ① N_out ⟵ Check_Norm_Contradiction(N^t_injected)
  N ⟵ concat(N_out, N)
  ② foreach g_i ∈ G do
      ⟨t_ci, f_si⟩ ⟵ g_i
      if ¬f_si(W^t) then
          N_i ⟵ {n ∈ N : n = ⟨r, g_i, ρ, ϕ, d⟩ ∧ ϕ(W^t) = true}
          for j ⟵ 1 to size(N_i) do
              ⟨r, g_i, ρ_j, ϕ_j, d_j⟩_[scope] ⟵ n_j
              if f_si = ρ_j then add(n_j, List_1)
              else if d_j = Obl ∨ d_j = Prohib then add(n_j, List_2)
          Ⓐ if card{N_i} = 0 then g'_i ⟵ ⟨t_ci, f_si⟩
          Ⓑ if card{N_i} = 1 then
              ⓘ if card{List_1} = 1 then
                ⟨r, g_i, ρ, ϕ, d⟩ ⟵ n
                if (d = Perm ∨ d = Obl) then t'_ci = OR_composition(t_ci, ϕ)
                else t'_ci = AND_composition(t_ci, ¬ϕ)
                g'_i ⟵ ⟨t'_ci, f_si⟩
              ⓘⓘ if card{List_2} = 1 then
                if t_ci(W^t) = true) then
                  f'_si ⟵ Const_Final_State(n, f_si)
                  g'_i ⟵ ⟨t_ci, f'_si⟩
          Ⓒ if card{N_i} > 1 then
              if card{List_1} ≥ 2 then
                List_1 ⟵ Chek_Ant(List_1, type_1) Case F
                List_1 ⟵ Chek_Ant(List_1, type_3) Case H
              if card{List_2} ≥ 2 then List_2 ⟵ Chek_Ant(List_2, type_1) Case C
              (ϕ_OR, ϕ_AND) ⟵ Comp_Norm_Cond(List_1)
              t'_ci ⟵ OR_composition(t_ci, ϕ_OR)
              t'_ci ⟵ AND_composition(t'_ci, ϕ_AND)
              f'_si ⟵ f_si; h ⟵ 1
              while h ≤ size(List_2) do
                ⟨r, g_i, ρ_h, ϕ_h, d_h⟩_[scope] ⟵ n_h
                cont ⟵ false; k ⟵ h + 1
                while k ≤ size(List_2) ∧ (cont = false) do
                  ⟨r, g_i, ρ_k, ϕ_k, d_k⟩_[scope] ⟵ n_k; k ⟵ k + 1
                  if ρ_h = ρ_k then
                    if (d_h = Obl ∧ d_k = Perm) ∨ (d_k = Obl ∧ d_h = Perm) then
                      f'_si = f'_si ∧ ρ_h Case D
                    if (d_h = Proh ∧ d_k = Perm) ∨ (d_k = Proh ∧ d_h = Perm)
                    then f'_si = f'_si ∧ ¬ρ_h Case E
                    cont ⟵ true
                    List_2 ⟵ remove(n, ρ = ρ_h, List_2)
                if cont = false then
                  f'_si ⟵ Constr_Final_State(n_h, f'_si); h ⟵ h + 1
              g'_i ⟵ ⟨t'_ci, f'_si⟩
          if t'_ci(W^t) = true then
              pursue(g'_i)
```

Algorithm 1: Norm compliance.

goal activation and the system cannot pursue that goal as long as the norm is applicable (i.e., $\phi(\mathscr{W}^t) = true$).

(ii) Step ⓘⓘ: $f_s \neq \rho$ then a new constrained final state is determined (see Algorithm 3) and the system tries to achieve such constrained final state.

The last situation (step Ⓒ) is the general case in which norms are more than one, and they can have different deontic operators. In this case, Algorithm 1 allows modifying goals, making them norm compliant according to a set of norms. Because norms are more than one, the presence of antinomy is possible. Thus, for each set of norms, antinomies are

Data: a list of norms $\mathcal{N}$
Result: a list $\mathcal{N}_{out}$ of consistent norms
$\mathcal{N}_{out} \longleftarrow \varnothing$
① **for** $i \longleftarrow 1$ **to** $length(\mathcal{N})$ **do**
  $\langle r, g, \rho, \phi_i, d\rangle_{[scope]} \longleftarrow n_i$
  $\langle t_c, f_s\rangle \longleftarrow g$
  if $(\phi \neq \bot)$ **then**
    if $(f_s \wedge d(\rho) \neq \bot) \vee (\phi_i \neq \top)$ **then**
      add n_i to $\mathcal{N}_{out}$
    else
      if $f_s = \rho$ **then**
        $alert(n_i)$ (see Case A)
      else
        $revise(n_i)$ (see Case B)
  else
    $revise(n_i)$

Algorithm 2: Check_Norm_Contradiction.

Data: an applicable norm n, a final state f_s
Result: a constrained final state f_s'
$\langle r, g, \rho, \phi, d\rangle \longleftarrow n$
switch d **do**
  case *Permission*
    $f_s' \longleftarrow f_s \wedge (\rho \wedge \neg\rho)$
  case *Prohibition*
    $f_s' \longleftarrow f_s \wedge \neg\rho$
  case *Obligation*
    $f_s' \longleftarrow f_s \wedge \rho$

Algorithm 3: Constrain_Final_State.

checked (see Algorithm 4). If $f_s \neq \rho$ the recovery is applied only for Case C. Conversely, if $f_s = \rho$ the recovery is applied for Case F and Case H. Hence, Algorithm 1 firstly works on norms where $f_s = \rho$ that modify goal's trigger condition. Indeed, by encapsulating the condition expressed by the norms into the goal they refer, it is possible to modify the activation of that goal thus making it compliant with the norms.

Such composition (see Algorithm 5) takes into consideration different types of norms, and it accordingly modifies the activation of a goal. Secondly, Algorithm 1 works on the set of norms where $f_s \neq \rho$ that modify goal's final state. In particular, if such set of norms contains at least two norms, Algorithm 1 checks if Case D or Case E exists and in such case it accordingly modifies the resulting final state to comply with such norms, and it removes from the list all norms that refer to the analysed case. Otherwise, it constrains the final state of the goal with the current norm according to Algorithm 3. In the following, we detail the algorithms that are invoked by the Algorithm 1.

6.1. Check Norm Contradiction Algorithm. Check Norm Contradiction Algorithm (see Algorithm 2) allows detecting incompatibility and inconsistent norms. In particular, Algorithm 2, for each norm, firstly checks the applicability condition to verify that it does not contain a logical contradiction (i.e., $\phi \neq \bot$). If it is the case, the norm has to be revised. This control is useful for avoiding the useless work of the system in evaluating some norms that will always be inapplicable. The second check of the Algorithm 2 (i.e., $(f_s \wedge d(\rho) \neq \bot) \wedge (\phi \neq \top)$) allows avoiding norms that are always in contradiction with the goal they refer to. Therefore, norms prevent the fulfillment of a goal in any state of the world. During this control, two different cases are highlighted (i.e., $f_s = \rho$ and $f_s \neq \rho$). If $f_s = \rho$, typically the norm is a prohibition for pursuing a goal (Case A). Because this norm could be deliberately introduced in the system to block a particular functioning, a simple alert is notified. If $f_s \neq \rho$ and $f_s \cap \rho \neq \varnothing$, this means that the norm affects only a portion of the final state of the world of the goal (Case B). Because in this case, the system would try to reach a goal which effectively cannot be reached creating a system overflow, a norm revision is mandatory.

6.2. Constrain Final State Algorithm. Algorithm 3 allows modifying a final state f_s according to the constraints expressed by a norm n. In particular, if norm n is a permission the constrained final state can or cannot include the state of affair the norm disciplines (i.e., $f_s' = f_s \wedge (\rho \wedge \neg\rho)$). If norm n is a prohibition the constrained final state can not include the state of the affair the norm disciplines (i.e., $f_s' = f_s \wedge \neg\rho$) and the contrary in the case of an obligation norm (i.e., $f_s' = f_s \wedge \rho$).

6.3. Check Antinomy Algorithm. When there is more than an applicable norm in the system, it is necessary to check for deontological contradictions among norms and remove them (if any). In our context, antinomy could be generated by different norms relating to the same goal. For instance, if there is an applicable norm $n1$ that prohibits to pursue a goal $g1$ and another applicable norm $n2$ that obliges to pursue the same goal $g1$, then $n1$ and $n2$ generate an antinomy. According to Section 5.3, Algorithm 4 manages three types of antinomy. We adopt the *legis posterior* and *legis superior* criteria coming from the legal theory. By adopting the *legis posterior* criterion, the most recent norm takes precedence. Conversely, by adopting the *legis superior*, the norm imposed by the prominent institutional power takes precedence. As we previously said, norms for different scopes could be defined. Norms for user well-being, for emergency management, and for ordinary situations could come in force. Some scopes are more crucial compared to other ones; a weight is applied for determining the relevance of a norm. We assign a priority to each norm taking into account the scope's weight and the publication date of the norm in the system, as follows.

Definition 11 (priority). Let a norm $n = \langle r, g, \rho, \phi, d\rangle_{[scope]}$. Let w_{scope} be the weight of the scope and let w_{time} be the weight of the time. The priority of n is

$$P_n = w_{scope} + w_{time} \tag{22}$$

```
Data: a list of applicable norms 𝒩, Type of Antinomy
Result: a list 𝒩_out of consistent norms
switch Type do
    case Type_1
        d_A = Prohibition
        d_B = Obligation
    case Type_2
        d_A = Obligation
        d_B = Permission
    case Type_3
        d_A = Prohibition
        d_B = Permission
① 𝒩_out ⟵ priority_order(𝒩)
ℳ_conflicts ⟵ ∅
② for i ⟵ 1 to length(𝒩_out) do
    ⟨r, g, ρ, ϕ_i, d_i⟩ ⟵ n_i
    ℳ_conflicts[i][i] ⟵ 1
    for j ⟵ i + 1 to length(𝒩_out) do
        if r_i = r_j ∧ ρ_i = ρ_j ∧ d_i ≠ d_j ∧ ((d_i = d_A ∧ d_j = d_B) ∨ (d_i = d_B ∧ d_j = d_A)) then
            ℳ_conflicts[i][j] ⟵ 1
        else
            ℳ_conflicts[i][j] ⟵ 0
ℳ_conflicts[j][i] ⟵ ℳ_conflicts[i][j]
𝒩'_out ⟵ 𝒩_out
③ for i ⟵ length(𝒩'_out) to 1 do
    n_current ⟵ 𝒩'_out[i]
    for j ⟵ i − 1 to 2 do
        if ℳ_conflicts[i][j] = 1 then
            delete(𝒩_out, n_current)
            break;
```

Algorithm 4: Check_Antinomy.

```
Data: a list of norms NormList
Result: a couple (ϕ_mergedOR, ϕ_mergedAND)
List ϕ_OR ⟵ ∅
List ϕ_AND ⟵ ∅
// Identification of norm types
for j ⟵ 1 to size(NormList) do
    ⟨r, g, ρ, ϕ, d⟩ ⟵ NormList[j]
    switch d do
        case Obligation
            break
        case Prohibition
            add ¬ϕ to List ϕ_AND
        case Permission
            add ϕ to List ϕ_OR
// Permissions give alternatives (OR)
if Size(List ϕ_OR) ≠ 0 then
    ϕ_mergedOR ⟵ List ϕ_OR[1]
    for h ⟵ 2 to Size(List ϕ_OR) do
        ϕ_mergedOR ⟵ OR_composition(ϕ_mergedOR, List ϕ_OR[h])
// Prohibition are mandatory (AND)
if Size(List ϕ_AND) ≠ 0 then
    ϕ_mergedAND ⟵ List ϕ_AND[1]
    for h ⟵ 2 to Size(List ϕ_AND) do
        ϕ_mergedAND ⟵ AND_composition(ϕ_mergedAND, List ϕ_AND[h])
```

Algorithm 5: Compose_Norm_Condition.

For removing conflicts, Algorithm 4 generates a conflicts matrix $\mathcal{M}_{conflicts}$ where $\mathcal{M}_{conflicts}[i][j] = 1$ indicates a conflict between i^{th} and j^{th} norm (step ②). Such a matrix is built starting from an ordered list of norms according to their priority (step ①). Hence, norms with the lowest priority are removed (step ③).

6.4. Compose Norm Condition Algorithm. Compose Norm Condition Algorithm, taking in input a list of norms related to a single goal g where for each norm $\rho = f_s$, allows merging in a unique activation condition, the conditions expressed by each norm. The algorithm takes into consideration different types of deontic operators. In the case of permissions, the resulting merged condition is an OR chain obtained by the single norm conditions, which relaxes the activation condition of the goal the norms refer to. Conversely, in the case of prohibitions, the resulting merged condition is an AND chain obtained by negating the conditions of the norms, which constrains the activation condition of the goal the norms refer to.

In the following, the approach is evaluated using the well-known case study in the field of the business rules.

7. Eurent Case Study

The Eurent scenario is a widely known case study adopted in the field of business processes for demonstrating the capabilities of the proposed solution. It was initially developed by Model System, Ltd. Briefly, the case study presents a car rental company with branches in several countries which provides common rental services. It also owns information about cars, branches, employees, and so on. Particular attention is paid to the information about the customers that allows establishing if they are good or bad clients. Each branch of the company owns a predetermined number of cars that are available for rental. A customer may rent a car through an advanced reservation or a walk-in rental. In an advanced reservation, the rental period and the car group are specified at the time of reservation. Conversely, walk-in rentals (i.e., immediate reservations) are also accepted if cars are available. When cars are returned, the renting branch has to ensure that it is returned at the end of the rental period. Moreover, a customer can have several reservations but only one car rented at the time. The company keeps information about customers, their rentals, and bad experiences such as late return, problems with payments, and damage to cars. The Eurent company adopts several rules for governing their business.

Some rules are related to the driver. For example, each customer must have a valid driver's license for driving the car and (s)he has to be insured. Some rules are defined for rental reservation acceptance. For example, if the customer requesting the rental has been blacklisted, the rental must be refused. A customer may have multiple future reservations but may have only one car at any time. In the case of advance reservations, if the requested model is not available, a car in the same group as the request model has to be assigned. If there are several available cars of the model requested, the one with the lowest mileage has to be assigned and so on. A small portion of a conceptual schema of the Eurent case study is shown in Figure 4. It defines the domain of knowledge used in the following. The proposed approach has been implemented in a simulated intelligent system able to manage car reservations. Figure 5 shows a portion of the system goal model. We adopt the following template for exemplifying a goal $goal(t_c, f_s)_{[Id]}$, where t_c is the trigger condition of the goal, f_s is the final state to be reached, and Id is an identification code for the goal. In the following, a small subset of Eurent goals is as follows:

g_0: `goal(start(customer,rental_agreement), or([state(rental_agreement,closed), state(rental_agreement,open), state(rental_agreement, cancelled)])`

g_1: `goal(requested(customer,rental_agreement), done(make_reservation))`

g_2: `goal(cancel_requested(customer,rental_agreement), done(cancel_reservation))`

The first one means that when the system is committed to managing reservation, the system tries to reach a state of the world where one of three possible final states is verified. The choice depends on the current state of the world the system is. In the following, it is shown a small subset of norms related to a g_2 where $\rho = f_s$ (i.e., N_1, N_2, N_3) and $\rho \neq f_s$ (i.e., N_4, N_5):

N_1: `norm(role(_), `g_2` , done(cancel_reservation), is_blacklisted(customer,true)),type(obligation))`

N_2: `norm(role(customer), `g_2` , done(cancel_reservation), state(car,assigned),type(permission))`

N_3: `norm(role(customer), `g_2` , done(cancel_reservation), state(car,in_rent)),type(prohibition))`

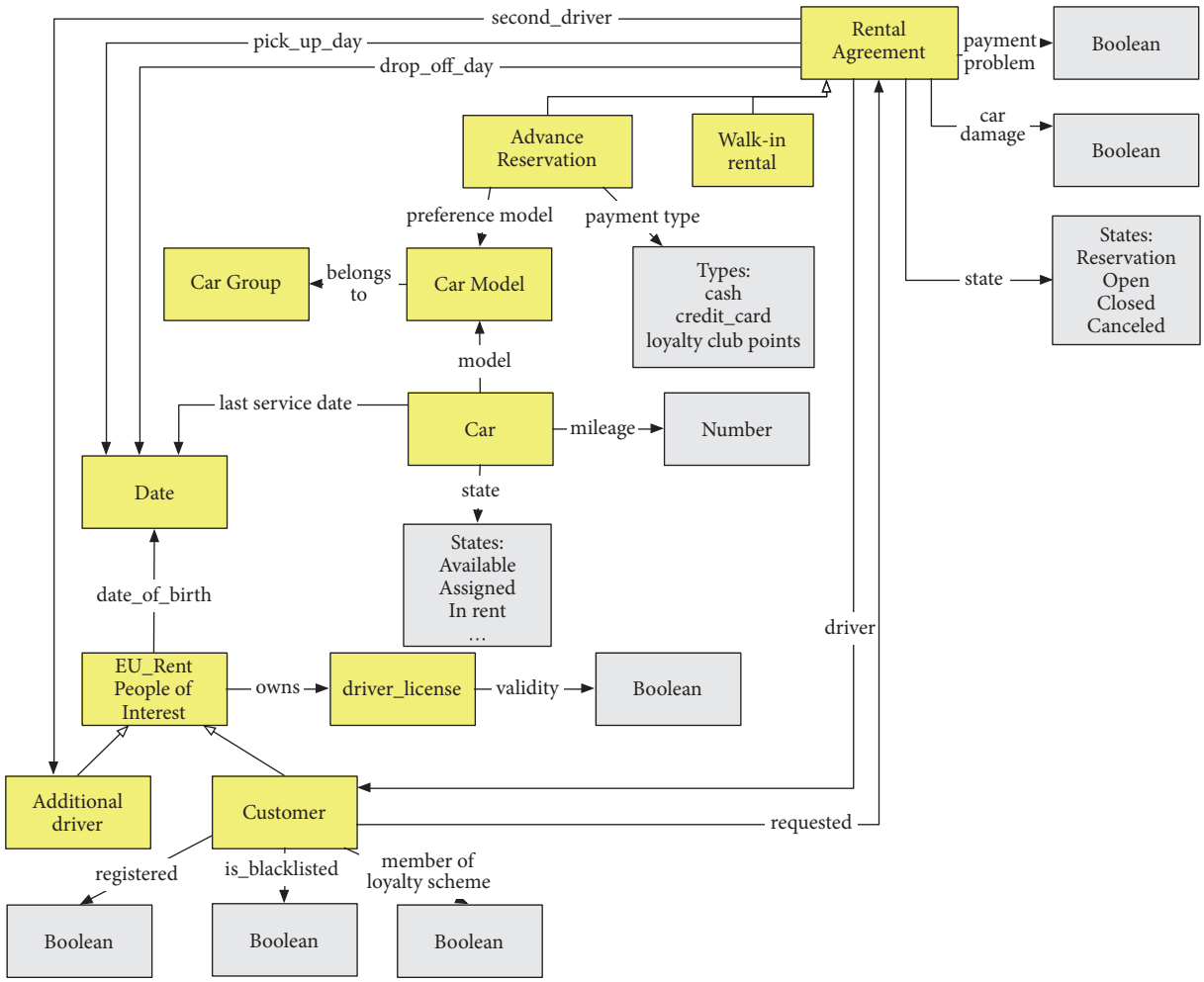

Figure 4: A portion of Eurent domain.

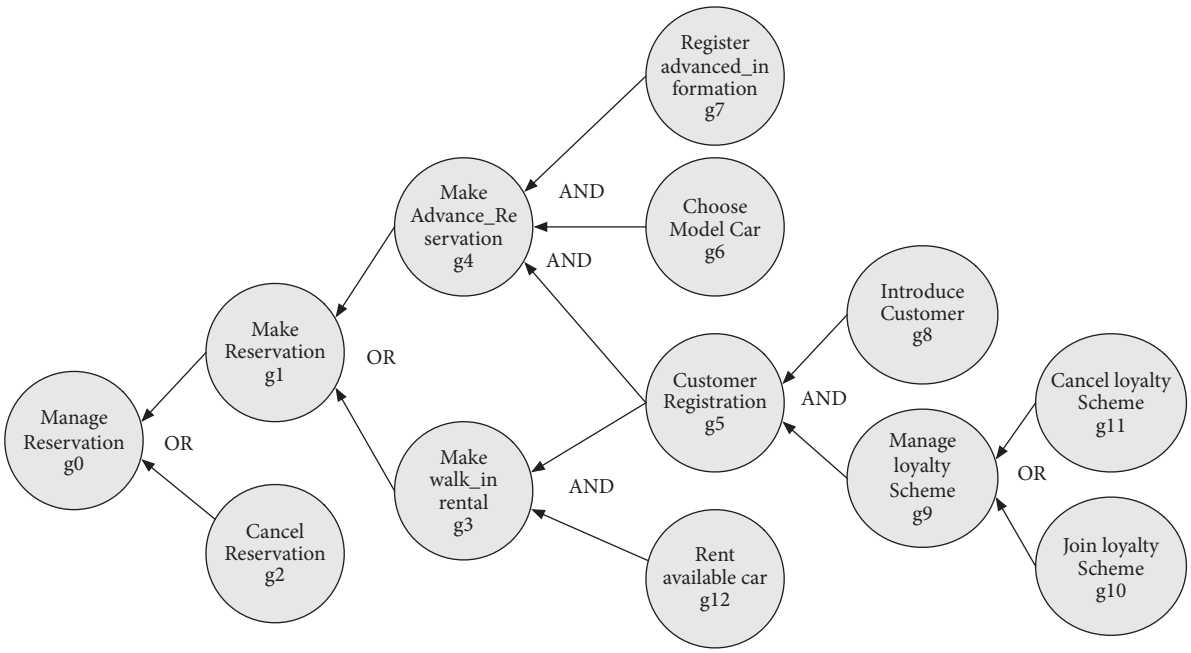

Figure 5: A portion of Eurent goal model.

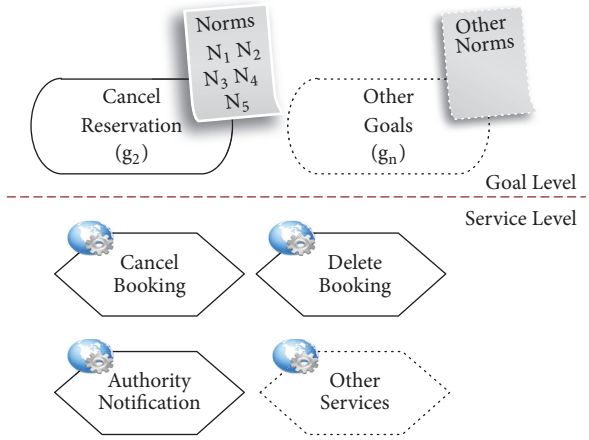

Figure 6: A subset of available services for Eurent case study.

N_4: `norm(role(_), `g_2` , activity(cancel(record)),`
`is_blacklisted(customer,true),type( prohibition))`

N_5: `norm(role(_),` g_2 `, activity(signal(authority)),`
`and([owns(customer,driver_license),`
`validity(driver_license,false)]),type(obligation))`

In particular, N_1 states if a customer is placed on a blacklist then all his bookings must be cancelled. N_2 states that it is permitted to cancel a reservation (without penalties) if a car has been only assigned to the customer. In this case, the customer is charged with no cost. Conversely complying to N_3, it is prohibited to cancel a reservation (without penalties) if the car is in rent (the state of the car is changed from *assigned* to *in rent* the day of the pick-up). In this case, one day-rental will be charged. N_4 differently states that it is prohibited to cancel the record related to a blacklisted customer. This norm avoids making erroneous cancellations from the organization database of blacklisted customers. Finally, N_5 imposes the obligation on the rental companies to alert the authority if a customer has no valid driver license. Figure 6 highlights some available services the system may use for satisfying its goals. For example, according to Eurent case study, two services allow the system for satisfying g_2, although they produce a different state of the world. In particular, the service *Cancel Booking* cancels only the customer's reservation. Conversely, the service *Delete Booking* deletes not only the reservation but also all the data relating to the customer. Moreover, in Figure 6 we also emphasize *Authority Notification* Service that will be used by the system for informing the authority about unlawful behaviours of the customers.

Each simulated scenario presents *(i)* an initial state of the world (W^0) that determines the applicability of some norms and the activation of some goals, *(ii)* a set of norms, *(iii)* the steps of the normative reasoning the system performs, and, finally, *(iv)* a brief analysis of the behaviour of the system. For the sake of simplicity, we isolated only the reasoning about goal g_2.

Scenario 1.

```
INITIAL CONDITIONS:
W^0 =cancel_requested(customer,rental_agreement), t_c(W^0) =true
APPLICABLE NORMS: No applicable norms
NORMATIVE REASONING:
[system] Trying to achieve g2
[system] Found available service: cancel_booking_service
[system] EXECUTING STEP A: NO NORMS for g2
[system] ...invoking cancel_booking_service
[monitor] The project is correctly terminated!
```

Scenario 1 shows the most simple situation, where no norm constrains the system. Thus, no change occurs in the normal behaviour of the system. The system pursues the triggered goal (g_2) by executing the initially selected service.

Scenario 2.

```
INITIAL CONDITIONS:
W^0 = {is_blacklisted(customer, true)} , t_c(W^0) =false
APPLICABLE NORMS:
N_1 = norm(role(customer), g_2 , done(cancel_reservation),
      is_blacklisted(customer,true)),type(obligation))
NORMATIVE REASONING:
[system monitor] Injected N_1
[system] SKIP STEP A
[system] EXECUTING STEP Bi: ONE NORM (Permission or Obligation) where RHO=FS for g_2
[system] Trying to achieve g_2
[system] Found available service: cancel_booking_service
[system] ...invoking cancel_booking_service
[monitor] The project is correctly terminated!
```

Without normative constraints, the system pursues a goal when its trigger condition is verified. By introducing a norm that obligates to obtain some state of affairs, the system is forced to pursue that goal when the condition of the obligation is verified although the trigger condition is not verified yet.

Scenario 3.

```
INITIAL CONDITIONS:
W^0 = {state (car,in_ rent)} , t_c(W^0) = false
APPLICABLE NORMS:
N_3 = norm(role(customer), g_2 , done(cancel reservation),
      state(car,in_rent)),type(prohibition))
NORMATIVE REASONING:
[system monitor] Injected N_3
[system] SKIP STEP A
[system] EXECUTING STEP Bi: ONE NORM (Prohibition) where RHO=FS for g_2
[system] SKIP STEP A
[system] EXECUTING STEP Bi: ONE NORM (Prohibition) where RHO=FS for g_2
    ...
```

When the trigger condition of a goal is false, the presence of a norm that prohibits pursuing that goal has no effects on the system behaviour. According to Definitions 4 and 5, such norm is applicable but not active. The system waits for any change in its state of the world.

Scenario 4.

```
INITIAL CONDITIONS:
W^0 = {state (car,in_rent), cancel_requested(customer,rental_ agreement)} , t_c(W^0) = true
```

```
APPLICABLE NORMS:
N3 = norm(role(customer), g2 , done(cancel reservation),
      state(car,in_rent)),type(prohibition))
NORMATIVE REASONING:
[system] Trying to achieve g2
[system] Found available service: cancel_booking_service
[monitor] Injected N3
[system] SKIP STEP A
[system] EXECUTING STEP Bi: ONE NORM (Prohibition) where RHO=FS for g2
[system] REPLANNING...
    ...
```

In this case, the presence of an active norm that prohibits pursuing a goal (g_2) nullifies the commitment of the system with that goal, thus causing a system replanning. As a result, after the replanning, the goal (g_2) is deleted by the set of the committed goals.

Scenario 5.

```
INITIAL CONDITIONS:
W^0 = {is_blacklisted(customer,true), cancel_requested(customer,rental_ agreement)} , t_c(W^0)
= true
APPLICABLE NORMS:
N4 = norm(role(_), g2 , activity(cancel( record)),
    is_blacklisted(customer,true),type(prohibition))
NORMATIVE REASONING:
[system] Trying to achieve g2
[system] Found available service: delete_booking_service
[monitor] Injected N4
[system] SKIP STEP A
[system] SKIP STEP Bi
[system] EXECUTING STEP Bii: ONE NORM (any type) where RHO/==FS for g2
[system] REPLANNING...
[system] Trying to achieve g2'
          g2' =goal(condition(cancel_requested(customer,rental_agreement)),
          condition(and([done(cancel_reservation), neg(cancel( record))])),
[system] Found available service: cancel_booking_service
[system] EXECUTING STEP A, NO NORMS for g2'
[system] ...invoking cancel_booking_service
[monitor] The project is correctly terminated!
```

To pursue the goal g_2, the system can invoke two services, Cancel or Delete Booking. As we previously said, the first one allows cancelling only the reservation. The second one allows also deleting the customer details. Thus, the system initially chooses the Delete Booking to pursue g_2. At run-time, a new norm that prohibits cancelling customer details if the customer is blacklisted is injected. Thus, to be compliant with the new norm, the system replans its goals. The effect of such norm is a change in the final state of the goal g_2, as can be seen after the replanning. Hence, to pursue g_2',

the system chooses the service Cancel Booking, and because in the current state of the world the trigger condition of such goal is verified, the system executes the step A because after replanning no other norms have been introduced. As a result, the system has achieved the desired state of affair compliant with the normative in force.

Scenario 6.

```
INITIAL CONDITIONS:
W^0 = {is_blacklisted(customer,true), owns(customer,driver_license), validity(driver_license,
      false)} t_c(W^0) = false
APPLICABLE NORMS:
N_1: norm(role(_), g_2 , done(cancel_reservation), is_blacklisted(customer,true)),type
   (obligation))
N_4: norm(role(_), g_2 , activity(cancel(record)), is_blacklisted(customer,true),type
   (prohibition))
N_5: norm(role(_), g_2 , activity(signal(authority)), and([owns(customer,driver_license),validity
   (driver license,false)]),type(obligation))
NORMATIVE REASONING:
[monitor] Injected N_1 , N_4 , N_5
[system] SKIP STEP A
[system] SKIP STEP Bi
[system] SKIP STEP Bii
[system] EXECUTING STEP C: ONE OR MORE NORMS (permission or obligation) where RHO==FS, NORMS (any
      type) where RHO/==FS for g_2
[system] REPLANNING...
[system] Trying to achieve g'_2
      g'_2 =goal(condition(is_blacklisted(customer,true)), condition(and([and([done(cancel_
      reservation),signal(authority)]), neg(cancel(record))]))
[system] Found available service: cancel_booking_service, authority_notification_service
[system] EXECUTING STEP A, NO NORMS for g'_2
[system] ...invoking authority_notification_service
[system] ...invoking cancel_booking_service
[monitor] The project is correctly terminated!
```

In this scenario the run-time injection of three norms applicable simultaneously is simulated. The effect of the first one N_1 is to force the system to cancel reservations made by blacklisted customers although nobody has demanded the system to commit with g_2 (indeed $t_c(W^t)$=false). For obtaining such an effect, the trigger condition of g_2 is changed. Conversely, the other two norms (N_4 and N_5) act on the final state of g_2 to lead the system in an admissible state of the world. As can be seen, after the replanning the system tries to achieve the new goal g'_2 by selecting two services that allow the system to reach the new constrained final state. It is worth noting that our approach does not change the practical reasoning of common goal-oriented systems. Our approach modifies the elements on which the system performs the practical reasoning, thus leading the system to make appropriate choices, both about what goals to deliberate and what means to choose for achieving the deliberated goal.

Scenario 7.

```
INITIAL CONDITIONS:
W^0 = {is_blacklisted(customer,true),is_memberof(eurent,true), cancel_requested(customer,
      rental_agreement)} t_c(W^0) =true
```

```
APPLICABLE NORMS:
N4: norm(role(_), g2 , activity(cancel(record)), is_blacklisted(customer,true),type
    (prohibition))
N6: norm(role(_), g2 , activity(cancel(record)), is_memberof(eurent,true),type(obligation))
NORMATIVE REASONING WITHOUT ANTINOMY DETECTION:
[monitor] Injected N6
[monitor] Injected N4
[system] SKIP STEP A
[system] SKIP STEP Bi
[system] SKIP STEP Bii
[system] EXECUTING STEP C: TWO OR MORE NORMS (any Type) with RHO/==FS for g2
[system] REPLANNING...
[system] Trying to achieve g2'
        g2' =goal(condition(cancel_requested(customer,rental_agreement)),condition(and([and
        ([done(cancel_reservation),cancel(record)]), neg(cancel(record))]))
[system] ERROR solutions not found ...
NORMATIVE REASONING WITH ANTINOMY DETECTION:
[monitor] Injected N6
[monitor] Injected N4
[system] SKIP STEP A
[system] SKIP STEP Bi
[system] SKIP STEP Bii
[system] EXECUTING STEP C:TWO OR MORE NORMS (any Type) where RHO/==FS for g2
[system] ANTINOMY DETECTED ... REMOVED N6
[system] REPLANNING...
[system] Trying to achieve g2'
        g2' =goal(condition(cancel_requested(customer,rental agreement)),condition(
        and([done(cancel_reservation),neg(cancel( record))])),
[system] Found available service: cancel_booking_service
[system] EXECUTING STEP A, NO NORMS for g2'
[system] ...invoking cancel_booking_service
[monitor] The project is correctly terminated!
```

In this scenario, we introduced a new norm N_6 that is not defined in the original Eurent case study. Its purpose is to simulate the presence of an anomalous situation. Such a norm is contrary with respect to N_4.

```
N6: norm(role(_),g2, activity(cancel(record)),
    is_memberof(eurent,true),type(obligation))
```

In the first situation without antinomy detection, the system is not able to find an appropriate solution to maintain compliance with both norms. Conversely, in the second case, the system detects the antinomy and removes it. In doing so, the system deletes N_6 because it is before N_4 and pursues the new goal constrained only by N_4. In such simulated system, we assume without loss of generality that each norm refers to the same scope and the time to come into force is trivially considered to be the one in which the norm was injected into the system.

The scenarios we presented show some possible situations that can occur in a dynamic normative context, highlighting the strengths of the proposed approach that we can summarize as follows:

(i) *Working with a dynamic set of norms*: norms may change for several reasons. For example, for employing the system in different working environments regulated by different normative corpus, for managing unexpected situations, and so on. In the proposed approach, the ability of the system to work with a dynamic set of norms is realized by decoupling the norms from the execution level. Norms act to a higher level of abstraction (i.e., goal level) without any modification to the action level. Scenario 5 shows how the introduction of a new norm acts directly on the goal by modifying it to reach the state of the world the norm prescribes. No change occurs at the action level.

(ii) *Ensuring norm compliance at run-time*: working with a dynamic set of norms requires that the system has to be able to verify the compliance with new injected norms during its execution. Scenarios 2 and 4 provide two examples of run-time compliance. Precisely, we can see the different effects on the system of an obligation and a prohibition. To act according to an obligation means to do what the obligation prescribes. Conversely, behaving compliant with a prohibition means not perform what the norm forbids. Scenario 2 shows how the system changes its behaviour as a consequence of the introduction of an obligation during the execution, by deliberating that goal whose fulfillment leads the system to be compliant to the norm. Conversely, Scenario 4 shows how the system changes its behaviour nullifying that already deliberated goal whose achievement would lead to a violation of the prohibition. Finally, Scenario 5 shows how the system reacts to a prohibition that regulates not the final state of the world, but the way the system has to reach the goal. Indeed, although the system has already planned the behaviour to execute for satisfying the current goal, the introduction of the norm leads the system to adapt to this new situation replans its behaviour choosing a new service to comply with the new norm.

(iii) *Handling norm conflict*: working with a dynamic set of norms may cause anomalous situations as aforementioned. Our approach provides mechanisms to avoid system deadlocks. An example is provided by Scenario 7 that shows how the system can detect an antinomy, remove it, and replan its behaviour taking into consideration the most recent norm.

(iv) *Improving system flexibility*: system flexibility is commonly viewed as the ability of the system to adapt to changes. The proposed approach improves flexibility endowing the system with the ability to adjust its behaviour for ensuring norm compliance also for norms that have not been defined at design-time. Precisely, the system may pursue a goal not initially deliberated that is become mandatory by the introduction of a norm, or it may inhibit the fulfillment of prohibited goals. In other cases, the system changes its behaviour to comply with new norms by choosing different services to pursue its goals.

8. Conclusions

The increasing employment of artificial systems that perform autonomously relevant tasks in several contexts raises the problem to provide them with the normative reasoning to ensure that they behave lawfully.

In this paper, we presented a normative reasoning approach for addressing norm compliance in the context of open and goal-directed systems. They are systems conceived for also satisfying not designed user requirements. They may evolve by increasing the objectives they can fulfill and by discovering new ways of achieving them. The approach presented in this work ensures run-time compliance with a dynamic set of norms that may change during system execution. The reasoning algorithm starting from the knowledge about norms, goals, and the state of the world allows the system to make decisions in conformity with the normative context, thus ensuring norm compliance at a higher level of abstraction (i.e., goal level) compared to the actions level. Thus, the effect of a norm acting at a goal level spreads out at the lower level of actions. Hence, a norm at goal level avoids defining normative constraints for each possible actions that a system could perform for reaching an objective and it makes the system free to choose such actions for reaching not constrained goals.

The flexibility of the approach also lies in the possibility of changing the set of norms dynamically without system redesigning. This feature poses a new issue to be addressed related to the possibility of introducing conflicting norms. The proposed algorithm taking into consideration the simultaneous presence of multiple norms related to the same goal, thus determining their joined effect by checking and solving norm conflicts and inconsistencies during system execution.

We are working for enlarging the normative reasoning by introducing alethic operators (such as necessity and possibility) to consider also definitional norms that are directed to regulate the elements of the knowledge domain rather than the behavioural aspect. Currently, we are also moving in the context of the social robots where the conformity of their behaviour according to social norms is considered extremely important to be accepted in daily human life.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] H. Prakken, "On how ai & law can help autonomous systems obey the law: a position paper," *AI4J–Artificial Intelligence for Justice*, vol. 42, 2016.

[2] N. Criado, "Reasoning about norms within uncertain environments," in *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems 2011, AAMAS 2011*, pp. 1255-1256, Taiwan, May 2011.

[3] M. E. Bratman, D. J. Israel, and M. E. Pollack, "Plans and resource-bounded practical reasoning," *Computational Intelligence*, vol. 4, no. 3, pp. 349–355, 1988.

[4] P. Ribino, C. Lodato, and M. Cossentino, "Modeling business rules compliance for goal-oriented business processes," in *Proceedings of the Workshop on Enterprise and Organizational Modeling and Simulation*, pp. 83–99, Springer, 2014.

[5] L. Frias, A. Queralt Calafat, and A. Olivé Ramon, "Eu-rent car rentals specification," Technical Report LSI Research Report. LSI-03-59-R, 2003.

[6] F. Dignum, "Autonomous agents with norms," *Artificial Intelligence and Law*, vol. 7, no. 1, pp. 69–79, 1999.

[7] T. Agotnes, W. Van Der, J. Hoek, C. Sierra, and M. Wooldridge, "On the logic of normative systems," in *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*, pp. 1181–1186, 2007.

[8] M. Sergot, "Action and agency in norm-governed multi-agent systems," in *Engineering Societies in the Agents World VIII*, p. 54, Springer, 2007.

[9] M. Dastani, N. A. M. Tinnemeier, and J.-J. C. Meyer, "A programming language for normative multi-agent systems," *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, pp. 397–417, 2009.

[10] N. Alechina, M. Dastani, and B. Logan, "Programming norm-aware agents," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, International Foundation for Autonomous Agents and Multiagent Systems*, pp. 1057–1064, 2012.

[11] M. J. Kollingbaum and T. J. Norman, "A contract management framework for supervised interaction," in *Proceedings of the Working Notes of the 5th UK Workshop on Multi-Agent Systems UKMAS*, 2002.

[12] W. W. Vasconcelos, A. García-Camino, D. Gaertner, J. A. Rodríguez-Aguilar, and P. Noriega, "Distributed norm management for multi-agent systems," *Expert Systems with Applications*, vol. 39, no. 5, pp. 5990–5999, 2012.

[13] S. Zhang, N. Gu, and J. Yang, "An norm-driven state machine model for CSCW systems," *Expert Systems with Applications*, vol. 31, no. 4, pp. 800–807, 2006.

[14] F. Meneguzzi and M. Luck, "Norm-based behaviour modification in BDI agents," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems*, pp. 177–184, 2009.

[15] N. Tinnemeier, M. Dastani, and J.-J. Meyer, "Programming norm change," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems*, pp. 957–964, 2010.

[16] M. Knobbout, M. Dastani, and J. C. Meyer, "Reasoning about dynamic normative systems," in *Logics in artificial intelligence*, vol. 8761 of *Lecture Notes in Comput. Sci.*, pp. 628–636, Springer, Cham, 2014.

[17] J. Jiang, H. Aldewereld, V. Dignum, and Y.-H. Tan, "Compliance checking of organizational interactions," *ACM Transactions on Management Information Systems (TMIS)*, vol. 5, no. 4, 2015.

[18] W. Vasconcelos, M. J. Kollingbaum, and T. J. Norman, "Resolving conflict and inconsistency in norm-regulated virtual organizations," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, ACM*, p. 91, 2007.

[19] M. Esteva, W. Vasconcelos, C. Sierra, and J. A. Rodríguez-Aguilar, "Norm consistency in electronic institutions," in *Advances in Artificial Intelligence – SBIA 2004*, vol. 3171 of *Lecture Notes in Computer Science*, pp. 494–505, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[20] N. B. Cocchiarella, *Notes on Deontic Logic*, Retrieved November 20 2015.

[21] R. J. Wieringa and J.-J. Meyer, "Applications of deontic logic in computer science: a concise overview," *Deontic Logic in Computer Science*, pp. 17–40, 1993.

[22] C. E. Alchourrón, "Logic of norms and logic of normative propositions," *Logique et Analyse*, vol. 12, pp. 242–268, 1969.

[23] G. H. Von Wright, "Deontic logic," *Mind*, vol. LX, no. 237, pp. 1–15, 1951.

[24] J. Hansen, *Imperatives and Deontic Logic*, On the Semantic Foundations of Deontic Logic, 2008.

[25] R. Reiter, *On Closed World Data Bases*, Springer, 1978.

[26] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An agent-oriented software development methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, pp. 203–236, 2004.

[27] L. Braubach, A. Pokahr, D. Moldt, and W. Lamersdorf, "Goal representation for BDI agent systems," in *Proceedings of the Second International Workshop on Programming Multi-Agent Systems, ProMAS 2004*, pp. 44–65, Springer, July 2004.