# Indirect Normative Conflict
## *Conflict that Depends on the Application Domain*

Viviane Torres da Silva[1], Christiano Braga[2] and Jean de Oliveira Zahn[2]
[1]*IBM Research (on leave from Universidade Federal Fluminense), Rio de Janeiro, Brazil*
[2]*Computer Science Department, Universidade Federal Fluminense, Niterói, Brazil*

Keywords: Agents, Norms, Conflicts, Application Domain.

Abstract: Norms are being used as a mechanism to regulate the behavior of autonomous, heterogeneous and independently designed agents. Norms describe what can be performed, what must be performed, and what cannot be performed in the multi-agent systems. Due to the number of norms specified to govern a multi-agent system, one important issue that has been considered by several approaches is the checking for normative conflicts. Two norms are said to be in conflict when the fulfillment of one norm violates the other and vice-versa. In this paper, we formally define the concept of an indirect normative conflict as a conflict between two norms that not necessarily have contradictory or contrary deontic modalities and that may govern (different but) related behaviors of (different but) related entities on (different but) related contexts. Finally, we present an ontology-based indirect norm conflict checker that automatically identifies direct and indirect norm conflicts on an ontology describing a set of norms and a set of relationships between the elements identified in the norms (behavior, entity and context).

## 1 INTRODUCTION

Norms have been significantly used as a mechanism to regulate the behavior of autonomous, heterogeneous and independently design agents. Therefore, several approaches have focused on the identification and resolution of conflicts between pairs of norms. The majority (Kollingbaum et al., 2008b; Vasconcelos and Norman, 2009) check for *direct and simple* normative conflict between two norms that have contradictory deontic modalities, such as prohibition and permission, or contrary deontic modalities, such as prohibition and obligation, (Elhag et al., 1999), and that are defined in the same context governing the same behavior executed by the same entity. An example of such direct conflict is when one norm prohibits and another allows an agent to execute a given action in a given context.

Few approaches focus on the identification of *indirect* conflicts between pairs of norms. A normative conflict is called indirect when the norms in conflict regulate the behavior of (different but) related entities; govern (different but) related behaviors, or are defined in (different but) related contexts. In addition, an indirect conflict can occur even thought the norms in conflict do not have contradictory or contrary deontic modalities. In (Oren et al., 2008) the authors consider that two obligation norms are in conflict if the entity (whose behavior is being governed by the norms) is not able to execute both obligations at the same time. In (Kollingbaum et al., 2008a; Vasconcelos et al., 2007; Vasconcelos et al., 2009) two norms are considered to be in conflict not only when they govern the same behavior but also when they govern related ones. However, to the best of our knowledge, the following situations are not considered simultaneously:

1. Conflicts between norms that do not have contradictory or contrary deontic modalities;
2. Conflicts between norms that regulate the behavior of (different but) related entities; govern (different but) related behaviors, and are defined in (different but) related contexts.

In this paper we consider two norms to be in indirect conflicts if they fall within the situations described in 1 and 2. As an example of such a situation, let us suppose that there is a norm *obliging attendees to make silence during talks* and another *allowing students to ask questions during classes*. In the traditional or direct conflict identification approach, a conflict between these two norms would not be detected


Torres da Silva V., Braga C. and de Oliveira Zahn J..
Indirect Normative Conflict - Conflict that Depends on the Application Domain.
DOI: 10.5220/0005350304520461
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 452-461
ISBN: 978-989-758-096-3

since they do not mention contradictory or contrary deontic modalities and the norms are not applied to the same entities, behaviors and contexts. However, we can say that there is an indirect conflict between these norms if we consider relationships among elements identified in the norms. For instance, let us consider the following scenario: (i) a student is an attendee; (ii) a class is a talk; and (iii) to make silence is orthogonal to ask questions, i.e., actions "to make silence" and "to ask questions" cannot be executed by the same or related entities at the same time. The conflict between the norms is not explicit (or direct) but camouflaged by the relationships between the elements addressed by the norms. We call such a conflict *indirect*.

We also introduce OnCheckIn, an ontology-based normative conflict checker that identifies both direct and indirect conflicts between two norms. It receives as input an ontology describing the norms and the relationships on entities, behaviors and contexts. Considering relationships between contexts and entities, it applies a set of norm propagation rules that propagate regulation from a context (or entity) to other contexts (or entities) related to it. Next, the conflict checker considers the relationship between the behaviors being regulated by the propagated norms to identify conflicts. When norms are propagated, indirect conflicts are made explicit and easily identified through the analysis of the relationships between behaviors. Thus, OnCheckIn is able to detect indirect conflicts between norms through norm propagation by taking into account the relationships among norm elements, that is, contexts, entities and behavior.

Is worth noting that we do not aim at an exhaustive or complete set of relationships between contexts, entities and behaviors. We propose an extensible approach where other relationships can be defined together with their propagation and conflict rules.

The remainder of this paper is organized as follows. Section 2 formalizes the concepts of norm and regulation that we will use throughout this paper. Sections 3 and 4 formalize the relationships between contexts and entities and define the semantics of the norm propagation restrictions associated with these relationships. Section 5 discusses the correctness of the norm propagation approach by proving that norm propagation does not add or remove regulation. Section 6 describes the relationships between behaviors and the conflict rules defined to identify conflicts. Section 7 presents OnCheckIn, our ontology-based tool for direct and indirect normative conflict detection. Section 8 describes related work. Section 9 concludes this paper with final remarks and future work.

## 2 NORM AND REGULATION DEFINITION

Our norm definition essentially[1] follows (Figueiredo and Silva, 2011) where, after studying ten specifications and implementations of normative languages, the authors have identified common elements to describe a norm.

**Definition 1** (Norm)**.** *A norm is a tuple in the set Norm*

$$Norm = D \times C \times \{E \cup \{\_\}\} \times B \times Cnd \times Cnd$$

*where D denotes the deontic modalities (obligation (ob), prohibition (pr) and permission(pe)); C is the context where the norm is defined; E is the entity whose behavior is being regulated by the norm; B is the behavior being regulated (i.e., an action); and $C_n$ indicates the conditions that activate and deactivate the norm, respectively. The symbol "_" denotes that a given norm regulates all entities in a given context. The scope of a norm is defined as its context.*

The entity, whose behavior is being regulated, must fullfil the norm when executing in the context where the norm is being defined. Outside its context the norm is not valid. In this paper, we consider that a norm can be defined in the context of an organization $o \in O$ or of an environment $env \in Env$. Thus, the set of possible contexts are defined as $C = O \cup Env$. A norm regulates the behavior of an agent $a \in A$, an organization (or group of agents) $o \in O$ or a role $r \in R$. Agents, organizations and roles are elements of the set $E = A \cup R \cup O$.

The activation condition $acc \in Cnd$ and deactivation condition $dac \in Cnd$ state the occurrence of an event such as a date, the execution of an action, or the fulfillment of a norm. In this paper, we will consider events to be natural numbers since we are mainly interested in the order of events occurrences rather than its structure. Thus, we use simple mathematic symbols such as $\leq$ and $\geq$ to indicate that an event occurs before or after another in a formula such as $(\forall n \in Norm)(n.acc \leq n.dac)$ meaning that it should be true that, for all norms, the activation condition of a given norm $n$ should happen before the deactivation condition of it. We use the "dot notation", as in $n.acc$, to denote the appropriate projection of a given tuple, such as the activation condition of a norm.

[1]The main differences between our norm definition and the one presented in (Figueiredo and Silva, 2011) are: (i) we have suppressed the definition of sanctions and rewards since they do not influence on the checking of conflicts and (ii) we are considering that a norm cannot only restrict the execution of an action but also the achievement of a state, as will be discussed in Section 5.

A norm can implicitly identify the agents and organizations whose behaviors are being regulated. Some norms use the symbol "_" to indicate that the they are applied to all agents and organizations executing in the contexts identified in the norms. Other norms may refer to roles played by agents or organizations in order to indicate that they apply to all agents and organizations that play such roles. In order to be able to explicitly identify the entities whose behaviors are being regulated by those norms, we have defined the *regulation function*. Regulation is able to identify all entities whose behavior is being regulated by a norm based on the relationships between contexts and entities identified in the domain ontology.

**Definition 2** (Regulation)**.** *A regulation identifies the entities whose behavior is being regulated by a given norm. Its input parameters are one norm and the set of relationships defined between entities and contexts that are represented in the domain ontology.*

$$regulation : Norm \times Rel \rightarrow E$$

## 3 CONTEXT RELATIONSHIPS

In this Section, we recall two well-known (Wooldridge, 2000; Weyns et al., 2005; W3C, 2004; Hubner et al., 2007) relationships among contexts: inhabit, in Definition 3, and hierarchy, in Definition 5. Associated with each relationship, we define the semantics of *norm propagation constraints* by using First-order logic. By norm propagation we mean that if a norm is solely defined at the level of a context then there must exist norms specifying the same regulation at the level of the elements of the given context. In this Section, we define, for each context relationship, a propagation constraint that formalizes regulation propagation for the given context relationship.

**Definition 3** (Inhabit)**.** *Relationship* inhabit *relates an entity to the environment where it executes. The entities that execute in the context of an environment are agents and organizations.*

$$Organization = \mathcal{P}(Agent)$$
$$inhabit \subseteq Environment \times (Agent \cup Organization)$$

*where $\mathcal{P}$ is the powerset operation.*

The norm defined in the context of an environment regulates the behavior of the agents and organizations executing in such a scope. If a norm is defined in the scope of an environment and it does not identify the entity which behavior is being regulated, such a norm applies to all entities that inhabit the environment. In order to model such norm, it is only necessary to use symbol "_" instead of explicitly identifying the entities. Thus, if we apply the *regulation function* (see Definition 2) to such a norm, it will return all agents and organizations that inhabit the environment identified in the norm, what indicates that such a norm governs the behavior of such entities.

**Definition 4** (Inhabit Propagation)**.** *If a norm is defined solely in the context of an environment, that is, it does not mention a specific entity, then it applies to all entities that inhabit the given environment,*

$$\exists env(\langle d, env, \_, b, acc, dac\rangle \in Norm \rightarrow$$
$$\forall e(inhabit(env, e) \rightarrow$$
$$\langle d, env, e, b, acc, dac\rangle \in Norm))$$

*where $d \in D$,$env \in Env$, $e \in A \cup O$, $b \in B$, $acc \in Cnd$, $dac \in Cnd$.*

If there is an environment and a norm applied to such environment, all entities that inhabit such environment will be regulated by a norm with the same characteristics as the first one.

**Definition 5** (Hierarchy)**.** *Relationship* hierarchy *relates an organization (called superorganization) to another organization (called suborganization) executing in the context of the superorganization.*

$$hierarchy \subseteq Organization \times Organization$$

If a norm is defined in the scope of an organization and it does not make it explicit the entity whose behavior is being regulated, then such a norm applies to all suborganizations of such an organization, as formalized by Definition 6, and to all roles of such an organization, as formalized by Definition 10. Thus, if we apply the *regulation function* (see Definition 2) to such a norm, it will return all suborganizations and roles played the organization identified as the context of the norm. It indicates that such a norm governs the behavior of the suborganizations and roles.

**Definition 6** (Hierarchy Propagation)**.** *If a norm is defined in the context of an organization and does not mention a specific entity then it applies to all suborganizations of the given organization.*

$$\exists o_1(\langle d, o_1, \_, b, acc, dac\rangle \in Norm \rightarrow$$
$$\forall o_2(hierarchy(o_1, o_2) \rightarrow$$
$$\langle d, o_1, o_2, b, acc, dac\rangle \in Norm))$$

*where $d \in D$, $o_1 \in O$, $o_2 \in O$, $b \in B$, $acc \in Cnd$, $dac \in Cnd$.*

## 4 ENTITY RELATIONSHIPS

Entity relationships relate agents and organizations to the roles they can play and roles to the organizations

where they are defined. In this section, we formalize these relationships and the *propagation restrictions* related to such relationships.

**Definition 7** (Play)**.** *Relationship* play *relates an agent or suborganization to the roles that it can play in the context of a given organization.*

$$play \subseteq Organization \times (Agent \cup Organization) \times Role$$

When a norm is applied to a role, it regulates the behavior of the agents or organizations that play such role in the context where the norm was defined. Thus, if we apply the *regulation function* (see Definition 2) to such a norm, it will return all agents and organizations playing the role in such context, what indicates that such a norm governs the behavior of the agents and organizations.

**Definition 8** (Play Propagation)**.** *If a norm regulates a role in a given organization then it regulates the behavior of all agents and suborganizations that can play the same role in the given organization.*

$$\exists r(\langle d,o,r,b,acc,dac\rangle \in Norm) \rightarrow \\ \forall e(play(o,e,r) \rightarrow \langle d,o,e,b,acc,dac\rangle \in Norm))$$

*where $d \in D$, $o \in O$, $r \in R$, $b \in B$, $e \in A \cup O$, $acc \in Cnd$, and $dac \in Cnd$.*

If there is a role and a norm applied to such role in a given organization, all entities that play such role in such organization will be regulated by a norm with the same characteristics as the first one.

**Definition 9** (Ownership)**.** *Relationship* ownership *relates a role to an organization where the role is defined.*

$$ownership \subseteq Organization \times Role$$

If a norm is defined in the scope of an organization and does not specific the entity whose behavior should be regulated, such a norm applies to all roles defined in the scope of such an organization. Thus, if we apply the *regulation function* (see Definition 2) to such a norm, it will return all roles played in such an organization, what indicates that such a norm governs the behavior of all agents and organizations that play such roles.

**Definition 10** (Ownership Propagation)**.** *If a norm is applied to all entities in an organization then it must be applied to all roles played in the organization.*

$$\exists o(\langle d,o,\_,b,acc,dac\rangle \in Norm \rightarrow \\ \forall r(ownership(o,r) \rightarrow \\ \langle d,o,r,b,acc,dac\rangle \in Norm))$$

*where $d \in D$, $o \in O$, $r \in R$, $b \in B$, $acc \in Cnd$, and $dac \in Cnd$.*

# 5 CORRECTNESS OF NORM PROPAGATION

The aim of this section is to demonstrate that, given a set of norms, the norm propagation restrictions specified in Sections 3 and 4 *preserve conflicts* in the given set of norms, i.e., *they do not include or remove any normative conflict*. Essentially, the propagation restrictions only require a given set of norms to make explicit regulations that were otherwise implicit as they are defined at the level of composite elements such as contexts or entities. In other words, *no new regulation is created or removed by norm propagation*.

**Lemma 1** (Inhabit Propagation Correctness)**.** *Inhabit propagation preserves regulation.*

*Proof.* Let $N$ be a set of norms with $n \in N, n = \langle d,env,\_,b,acc,dac\rangle$, that is, $n$ is a norm defined solely in the context of the environment *env*. Therefore, by applying *regulation function* (see Definition 2), $n$ governs the behavior of all agents and organizations that inhabit *env*. By Definition 4, the propagation of $n$ requires norms $n_e = \langle d,env,e,b,acc,dac\rangle$, where $inhabit(env,e)$, also to be in $N$. The propagation of $n$ does not imply any new regulation in $N$ that was not already specified by $n$. The entities whose behaviors were regulated by $n$ are exactly the same entities whose behaviors are being regulated by the set of $n_e$. In addition, since $n_e$ is propagated based on $n$, its other parameters have not been changed, as shown in Definition 4. □

**Lemma 2** (Hierarchy Propagation Correctness)**.** *Hierarchy propagation preserves regulation.*

*Proof.* Let $N$ be a set of norms with $n \in N, n = \langle d,org,\_,b,acc,dac\rangle$, that is, $n$ is a norm defined solely in the context of an organization *org*. Therefore, by applying *regulation function* (see Definition 2), $n$ governs the behavior of all suborganizations of *org*. By Definition 6, the propagation of $n$ requires norms $n_s = \langle d,org,suborg,b,acc,dac\rangle$, where $hierarchy(org,suborg)$, also to be in $N$. The propagation of $n$ does not imply any new regulation in $N$ that was not already specified by $n$. The suborganizations whose behaviors were regulated by $n$ are exactly the same suborganizations whose behaviors are being regulated by the set of $n_s$. In addition, since $n_s$ is propagated based on $n$, its other parameters have not been changed, as shown in Definition 6. □

**Lemma 3** (Play Propagation Correctness)**.** *Play propagation preserves regulation.*

*Proof.* Let $N$ be a set of norms with $n \in N, n = \langle d, o, r, b, acc, dac \rangle$, that is, $n$ is a norm defined in the context of an organization $o$ and applied to role $r$. Therefore, by applying *regulation function* (see Definition 2), $n$ governs the behavior of all agents or organizations playing role $r$ in $o$. By Definition 8, the propagation of $n$ requires norms $n_e = \langle d, o, e, b, acc, dac \rangle$, where $play(o, e, r)$, also to be in $N$. The propagation of $n$ does not imply any new regulation in $N$ that was not already specified by $n$. The agents and organizations whose behaviors were regulated by $n$ are exactly the same agents and organizations whose behaviors are being regulated by the set of $n_e$. In addition, since $n_e$ is propagated based on $n$, its other parameters have not been changed, as shown in Definition 8. □

**Lemma 4** (Ownership Propagation Correctness)**.** *Ownership propagation preserves regulation.*

*Proof.* Let $N$ be a set of norms with $n \in N, n = \langle d, o, \_, b, acc, dac \rangle$, that is, $n$ is a norm defined in the context of an organization $o$. Therefore, by applying *regulation function* (see Definition 2), $n$ governs the behavior of all role $r$ played in $o$. By Definition 10, the propagation of $n$ requires norms $n_r = \langle d, o, r, b, acc, dac \rangle$, where $ownership(o, r)$, also to be in $N$. The propagation of $n$ does not imply any new regulation in $N$ that was not already specified by $n$. The roles governed by $n$ are exactly the same roles governed by the set of $n_e$. In addition, since $n_e$ is propagated based on $n$, its other parameters have not been changed, as shown in Definition 10. □

**Theorem 1** (Norm propagation correctness)**.** *Norm propagation does not add or remove regulation of a given set of norms.*

**Proof.** By lemmata 1 to 4. □

# 6 ACTION RELATIONSHIPS

The following subsections define four kinds of relationships used to relate actions. These relationships were selected because they are well-known in the planning literature and the majority has been formalized in TAEMS (Horling et al., 1999)[2], a modeling language used to describe task structures and problem-solving activities of intelligent agents in a multi-agent environment. As stated before, is not our goal to be exhaustive on identifying all possible relationships among actions. We propose an extensional approach where the set of relationships among actions can be conservatively extended, that is, preserving the semantics of the basic relationships defined here.

In this paper, we consider a conflict is represented as a predicate between two norms.

**Definition 11** (Conflict)**.** *A conflict is a predicate between two norms.*

$$conflict : Norm \times Norm \rightarrow \{true, false\}$$

To check for conflicts that occur due to the relationships among actions in the norms, we define action conflict rules among related actions. The conflict rules consider that all propagation restrictions (from Definitions 4, 6, 8 and 10) are true in a given set of Norm individuals. *Conflict rules are thus defined on pairs of norms where their contexts and the entities whose behavior is being regulated by the norms are the same.*

## 6.1 Action Refinement

Action refinement relates two actions: a subaction and a superaction. The execution of the subaction must achieve the goal of executing the superaction, and possibly other goals. If there is more than one subaction for a given superaction, the execution of some subaction must achieve the goal of executing the superaction. We have defined the conflict rules from the point of view of the superaction.

In order to give a simple example of action refinement, let's consider the actions *to move*, *to fly* and *to taxi*. The last two actions are subactions of the superaction *to move*, i.e., when a plane flies or taxis it is moving.

**Definition 12** (Refinement)**.** *Relationship* refinement *relates an action (called superaction) to another action (called subactions).*

$$refinement \subseteq Action \times Action$$

**Definition 13** (Refinement Conflict Rule for Prohibition)**.** *If there is a prohibition applied to a superaction and an obligation or permission applied to a subaction then these norms are in conflict.*

$$\begin{aligned} &\exists b_1 \exists b_2 ((\langle pr, c, e, b_1, acc_1, dac_1 \rangle \in Norm \wedge \\ &\langle d, c, e, b_2, acc_2, dac_2 \rangle \in Norm \wedge \\ &refinement(b_1, b_2)) \rightarrow \\ &conflict(\langle pr, c, e, b_1, acc_1, dac_1 \rangle, \\ &\qquad \langle d, c, e, b_2, acc_2, dac_2 \rangle)) \end{aligned}$$

*where* $d \in \{ob, pe\}$, $c \in C$, $e \in E$, $b_1$ *and* $b_2 \in B$, $acc_i$ *and* $dac_i \in Cnd$, $acc_1 \geq dac_2$, *and* $dac_1 \geq acc_2$.

If there is two behaviors $b_1$ and $b_2$, a norm prohibiting the execution of $b_1$, another norm obligating

[2] Refinement is equivalent to max, composition is equivalent to sum and dependency is equivalent to enables. Max, sum and enable were defined in TAEMS.

or permitting the execution of $b_2$, and the relationship refinement is defined between $b_1$ and $b_2$, then such norms are in conflict.

**Definition 14** (Refinement Conflict Rule Schema for Obligation and Permission)**.** *If there is an obligation or permission applied to a superaction and prohibitions applied to all subactions then these norms are in conflict.*

$$\exists b_1(\langle d,c,e,b_1,acc_1,dac_1\rangle \in Norm \rightarrow \forall b_i(refinement(b_1,b_i) \rightarrow \langle pr,c,e,b_i,acc_2,dac_2\rangle \in Norm \rightarrow conflict(\langle d,c,e,b_1,acc_1,dac_1\rangle, \langle pr,c,e,b_i,acc_2,dac_2\rangle)))$$

*where* $b_i \in \{b \mid refinement(b_1,b)\}$, $d \in \{ob,pe\}$, $c \in C$, $e \in E$, $b_1$ *and* $b_2 \in B$, $acc_i$ *and* $dac_i \in Cnd$, $acc_1 \geq dac_2$, *and* $dac_1 \geq acc_2$.

If there is a behavior $b_1$ and an obligation or prohibition applied to $b_1$, and if all behaviors $b_i$ that are refinements of $b_1$ have a norm prohibiting their execution, a conflict exists between the norm applied to $b_1$ and the one applied to $b_i$.

## 6.2 Action Composition

If a composition relationship is defined between two actions, then there is an action called part that is part of an action called whole and the whole action is an abstract action. In order to achieve the goals of executing the whole action it is necessary to execute all its parts. Since the norms defined to the whole action influence the execution of the part actions, we have defined the conflict rules from the point of view of the whole action.

Let's consider there is a need to govern a multiagent system. Such an action implies the execution of three other actions: to find out violations, to find out fulfillments and to apply sanctions. The action *to govern* is composed of these three actions.

**Definition 15** (Composition)**.** *Relationship* composition *relates an action (called whole) to other actions (called parts).*

$$composition \subseteq Action \times Action$$

**Definition 16** (Composition Conflict Rule for Obligation and Permission)**.** *If an obligation or permission is defined for a whole action and a prohibition is defined for a part action then there is a conflict.*

$$\exists b_1 \exists b_2((\langle d,c,e,b_1,acc_1,dac_1\rangle \in Norm \wedge \langle pr,c,e,b_2,acc_2,dac_2\rangle \in Norm \wedge composition(b_1,b_2)) \rightarrow conflict(\langle d,c,e,b_1,acc_1,dac_1\rangle, \langle pr,c,e,b_2,acc_2,dac_2\rangle))$$

*where* $d \in \{ob,pe\}$, $c \in C$, $e \in E$, $b_1$ *and* $b_2 \in B$, $acc_i$ *and* $dac_i \in Cnd$, $acc_2 \geq acc_1$,*and* $dac_1 \geq dac_2$.

**Definition 17** (Composition Conflict Rule Schema for Prohibition)**.** *If there is a prohibition applied to a whole action then a conflict will arise if all part actions are being obliged or permitted.*

$$\exists b_1(\langle pr,c,e,b_1,acc_1,dac_1\rangle \in Norm \rightarrow \forall b_i(composition(b_1,b_i) \rightarrow conflict(\langle pr,c,e,b_1,acc_1,dac_1\rangle, \langle d,c,e,b_i,acc_2,dac_2\rangle)))$$

*where* $b_i \in \{b \mid composition(b_1,b)\}$, $d \in \{ob,pe\}$, $c \in C$, $e \in E$, $b_1$ *and* $b_2 \in B$, $acc_i$ *and* $dac_i \in Cnd$, $acc_2 \geq acc_1$,*and* $dac_1 \geq dac_2$.

## 6.3 Action Orthogonality

If there exists an orthogonal relationship between two actions then both actions cannot be executed at the same time by the same entity. For instance, let's consider the action *to fly* and *to taxi*. A plane cannot execute these two actions at the same time.

**Definition 18** (Orthogonality)**.** *Relationship* orthogonal *relates two actions that cannot be executed simultaneously.*

$$orthogonal \subseteq Action \times Action$$

**Definition 19** (Orthogonality Conflict Rule)**.** *Orthogonal actions are in conflict if they are both obligations, or one is an obligation and the other a permission.*

$$\exists b_1 \exists b_2((\langle ob,c,e,b_1,acc_1,dac_1\rangle \in Norm \wedge \langle d,c,e,b_2,acc_2,dac_2\rangle \in Norm \wedge orthogonal(b_1,b_2)) \rightarrow conflict(\langle ob,c,e,b_1,acc_1,dac_1\rangle, \langle d,c,e,b_2,acc_2,dac_2\rangle))$$

*where* $d \in \{ob,pe\}$, $c \in C$, $e \in E$, $b_1$ *and* $b_2 \in B$, $acc_i$ *and* $dac_i \in Cnd$, $acc_1 \geq dac_2$ *and* $dac_1 \geq acc_2$.

In any other case, there is not a conflict. One must observe that a prohibition norm applied to an action $ac$ will never conflict with another norm applied to an action $ac'$ orthogonal to $ac$. If $ac$ is prohibited, the entity can execute $ac'$.

## 6.4 Action Dependency

If a dependency relationship is defined between two actions $ac_1$ and $ac_2$ then $ac_1$ must be executed before $ac_2$. If the client action is not executed, the dependent action cannot be executed. The state that is the precondition of the dependent action can only be achieve

by executing the client action, i.e., the effect of executing the client action is a subset of the precondition of the dependent action.

In order to exemplify such relationship, let's consider again the example of governing a multi-agent systems. The action *to applySanctions* depends on the action *to findOutViolations*.

**Definition 20** (Dependency)**.** *Relationship* dependent *relates an action (called client) to another (called dependent).*

$$dependent \subseteq Action \times Action$$

**Definition 21** (Dependency Conflict Rule)**.** *If there is a prohibition applied to any client action and an obligation or permission applied to a dependent action then these norms are in conflict.*

$$\begin{aligned}&\exists b_1 \exists b_2 ((\langle pr, c, e, b_1, acc_1, dac_1\rangle \in Norm \wedge \\ &\quad \langle d, c, e, b_2, acc_2, dac_2\rangle \in Norm \wedge \\ &\qquad dependent(b_1, b_2)) \rightarrow \\ &\qquad conflict(\langle pr, c, e, b_1, acc_1, dac_1\rangle, \\ &\qquad\quad \langle d, c, e, b_2, acc_2, dac_2\rangle))\end{aligned}$$

*where* $d \in \{ob, pe\}$, $c \in C$, $e \in E$, $b_1$ *and* $b_2 \in B$, $acc_i$ *and* $dac_i \in Cnd$, $acc_1 \geq dac_2$,*and* $dac_1 \geq acc_2$.

One must note that a prohibition applied to a dependent action will never conflict with any other norm applied to its client action.

# 7 CONFLICT CHECKER

In this Section we present the algorithms that implement OnCheckIn. As stated before in Section 1, our approach is based on an ontology (available at http://goo.gl/Qkutmo) that is used to describe the norms by following definition 1 and all relationships between contexts, entities and actions defined in previous sections. The *onCheckIn* algorithm assumes that the ontology has been already parsed in two groups, one containing all the norms and another all the relationships. Thus, the *onCheckIn* algorithm receives as parameters the set of norms and the set of relationships defined in the domain ontology. The algorithm starts by calling the propagation algorithms *contextPropagation* and *entityPropagation*.

Propagation algorithms implement the propagation restrictions defined in Sections 3 and 4 as rules that generate new norms from the ones defined at the level of composite elements such as an environment or organization. They assume that a given ontology defines only the norms applied to the more general contexts or entities. The ontology designer does not have to make explicit all the norms applied to more concrete contexts or entities. It is the aim of OnCheckIn to propagate the norms by following the relationships between contexts and entities.

After propagating the norms, for each pair of norms defined over the same context and that govern the behavior of the same entity, the *onCheckIn* algorithm checks if the norms are active at the same time, given their activation and deactivation conditions. If two norms are active, there may be a conflict if the behavior being governed is the same (in the case of direct conflicts) or if they are related by one of the action relationships described in Section 6 (in the case of indirect conflicts). The conflict rules are implemented in Algorithms 4 to 7.

Our approach can be easily extended in two directions:

1. other relationships between contexts and entities can be defined together with their propagation rules. It is necessary to extend Algorithms 1 and 2;
2. other action relationships can be created together with their conflict rules. It is necessary to define a new conflict rule algorithm to deal with each new relationship and include a call to this new algorithm in Algorithm 1.

**Algorithm 1:** OnCheckIn.

**Require:** N : set of norms, R : set of relationships

```
1:  function ONCHECKIN(N,R)
2:      N = CONTEXTPROPAGATION(N)
3:      N = ENTITYPROPAGATION(N)
4:      conflict = false
5:      for all n1 ∈ N do
6:          for all n2 ∈ N do
7:              if (n1.c = n2.c) ∧ (n1.e = n2.e) then
8:                  if intersect(n1,n2) then
9:                      conflict ←
10:                     n1.b = n2.b ∨
11:                     REFINEMENTCR(n1,n2,N) ∨
12:                     COMPOSITIONCR(n1,n2,N) ∨
13:                     ORTHOGONALITYCR(n1,n2) ∨
14:                     DEPENDENCYCR(n1,n2)
15:     return conflict
```

# 8 RELATED WORK

Checking normative conflicts is one of the main challenges on the development of normative multi-agent systems. The literature is abundant on the identification of such conflicts. Most of it focuses on checking simple and direct conflicts, such as (Kollingbaum

et al., 2008b; Vasconcelos and Norman, 2009). Since the identification of direct conflicts is not the focus of this paper, we will discuss in this Section related work on the identification of indirect conflicts.

**Algorithm 2:** ContextPropagation.

**Require:** N : set of norms
1: **function** CONTEXTPROPAGATION($N$)
2: **for all** $n \in N$ **do**
3: **if** $(n.c.type = \text{“}env\text{”}) \wedge (n.e = \emptyset)$ **then**
4: **for all** $e \in$ INHABIT$(n.c)$ **do**
5: $n_1 \leftarrow$
6: $\langle n.d, n.c, e, n.b, n.acc, n.dac \rangle$
7: $N =$ INCLUDENORM$(n_1, N)$
8: **if** $(n.c.type = \text{“}org\text{”})$ **then**
9: **for all** $o \in$ HIEARCHY$(n.c)$ **do**
10: $n_1 \leftarrow$
11: $\langle n.d, n.c, o, n.b, n.acc, n.dac \rangle$
12: $N =$ INCLUDENORM$(n_1, N)$
13: **return** $N$

**Algorithm 3:** EntityPropagation.

**Require:** N : set of norms
1: **function** ENTITYPROPAGATION($N$)
2: **for all** $n \in N$ **do**
3: **if** $(n.e.type = \text{“}role\text{”})$ **then**
4: **for all** $e \in$ PLAY$(n.c, n.e)$ **do**
5: $n_1 \leftarrow$
6: $\langle n.d, n.c, e, n.b, n.acc, n.dac \rangle$
7: INCLUDENORM$(n_1, N) \vee$
8: **if** $(n.c.type = \text{“}org\text{”}) \wedge (n.e = \emptyset)$ **then**
9: **for all** $r \in$ OWNERSHIP$(n.c)$ **do**
10: $n_1 \leftarrow$
11: $\langle n.d, n.c, r, n.b, n.acc, n.dac \rangle$
12: INCLUDENORM$(n_1, N) \vee$
13: **return** $N$

In (Cholvy and Cuppens, 1995; Gunay and Yolum, 2013) the authors describe that norms are applied to roles and that agents can play different roles. Similarly to our approach, they also consider a relationship called play between agents and roles when checking for conflicts between the norms.

In (Gaertner et al., 2007; Garcia-Camino et al., 2007) the conflict checker considers that a norm applied to an activity can be propagated to other activities. Moreover, (Garcia-Camino et al., 2007) takes into account the composition relationship between activities. Similarly, the conflict checker in (Kollingbaum et al., 2008a; Vasconcelos et al., 2007) allows for the composition relationship and also the delegation relationship. In (Vasconcelos et al., 2009), the

**Algorithm 4:** RefinementCR.

**Require:** N : set of norms, $n_1$ and $n_2$: two norms
1: **function** REFINEMENTCR($n_1, n_2$)
2: **if** $((n_1.d =' pr') \wedge (n_2.d = (\text{“}ob\text{”} \vee \text{“}pe\text{”})) \wedge$
3: REFINEMENT$(n_1.b, n_2.b))$ **then**
4: **return** *true*
5: **if** $((n_1.d = (\text{“}ob\text{”} \vee \text{“}pe\text{”})) \wedge (n_2.d =' pr') \wedge$
6: REFINEMENT$(n_1.b, n_2.b))$ **then**
7: **for all** $a \in$ SUBACTIONSOF$(n_1.b)$ **do**
8: **for all** $n \in N$ **do**
9: **if** $((n.b = a) \wedge$
10: $(n.d \neq \text{“}pr\text{”}))$ **then**
11: **return** *false*
12: **return** *true*
13: **return** *false*

**Algorithm 5:** CompositionCR.

**Require:** N : set of norms, $n_1$ and $n_2$: two norms
1: **function** COMPOSITIONCR($n_1, n_2$)
2: **if** $((n_1.d = (\text{“}ob\text{”} \vee \text{“}pe\text{”})) \wedge (n_2.d = \text{“}pr\text{”})) \wedge$
3: COMPOSITION$(n_1.b, n_2.b))$ **then**
4: **return** *true*
5: **if** $((n_1.d = \text{“}pr\text{”}) \wedge$
6: $(n_2.d = (\text{“}ob\text{”} \vee \text{“}pe\text{”}))) \wedge$
7: COMPOSITION$(n_1.b, n_2.b))$ **then**
8: **for all** $a \in$ PARTACTIONSOF$(n_1.b)$ **do**
9: **for all** $n \in N$ **do**
10: **if** $((n.b = a) \wedge$
11: $(n.d \neq (\text{“}ob\text{”} \wedge \text{“}pe\text{”})))$ **then**
12: **return** *false*
13: **return** *true*
14: **return** *false*

**Algorithm 6:** OrthogonalityCR.

**Require:** $n_1$ and $n_2$: two norms
1: **function** ORTHOGONALITYCR($n_1, n_2$)
2: **if** $((n_1.d =' ob') \wedge (n_2.d = ('ob' \vee' pe')) \wedge$
3: ORTHOGONAL$(n_1.b, n_2.b))$ **then**
4: **return** *true*
5: **return** *false*

**Algorithm 7:** DependencyCR.

**Require:** $n_1$ and $n_2$: two norms
1: **function** DEPENDENCYCR($n_1, n_2$)
2: **if** $((n_1.d = ('ob' \vee' pe')) \wedge (n_2.d =' pr') \wedge$
3: DEPENDENT$(n_1.b, n_2.b))$ **then**
4: **return** *true*
5: **return** *false*

same authors have included in the conflict checker the refinement relationship but they have only considered the simplest case when a superaction is specified by only one subaction. All these approaches take into account only the relationships between actions on the checking of indirect conflicts. We claim that they are incomplete since they ignore relationships among entities and contexts and that they are not able to find out conflicts between norms that do not define contradictory or contrary deontic modalities.

Besides considering the composition relationships between actions, the work in (Fenech et al., 2009) also consider that there may be norms applied to the same agent regulating contradictory or orthogonal actions. Similarly, in (Oren et al., 2008; Giannikis and Daskalopulu, 2009) the authors consider conflicts between obligations applied to the same agent that govern actions that cannot be executed at the same time. Although these approaches are able to consider conflicts between norms that have the same deontic modality, they do not consider the relationships between contexts and entities and they focus only on one or two relationships between actions.

Approaches such as (Dung and Sartor, 2011; Li et al., 2013a; Li et al., 2013b) focus on conflicts between norms defined in different contexts or in different domains. These works are not related to ours since we consider that the norms being checked are defined in the same multi-agent system.

## 9 CONCLUSION

This work presents the identification of *indirect* normative conflicts, that is, conflicts that can only be found when considering relationships of the domain. Not only direct conflicts between pairs of norms, but also indirect conflicts can be detected by the OnCheckIn algorithm which it takes into account the relationships among contexts, entities and behavior.

The relationships between contexts and entities considered by the algorithm have been formalized together with their propagation constraints. We have also formalized the relationships between actions and defined rules to identify conflicts. As explained in Section 7, our algorithm are *extensible* in order to accommodate other relationships, propagation constraints and conflict rules can be defined.

A preliminary version of the conflict checker (available at http://goo.gl/Qkutmo) was implemented in Java using the OWL API library (Horridge and Bechhofer, 2011). The user is able to load an OWL ontology defining the norms and the relationships. The conflict checker parsers the ontology and, for each pair of norms, checks for conflicts by considering the relationships between the elements of these norms. After the analysis, it informs the user about the pairs of norms in conflict and the reason(s) of the conflict, i.e., it informs if it is a direct conflict or an indirect conflict camouflaged by a set of relationships between the elements of the norms being analyzed. One small difference between this preliminary version and the OnCheckIn algorithm presented in this paper is that it is not based on the propagation of norms. Instead of propagating the norms according to the relationships between contexts and entities, it checks, for each pair of norms, if the contexts of the norms are related and if the entities are related. Then it checks if the behaviors are related. Although this approach consumes less memory space than the one presented in this paper (since it does not increase the set of norms), it is less efficient since it must check, for each pair of norms, if there are relationships between the contexts and relationships between the entities of these norms.

We are in the process of adapting the preliminary version of the conflict checker to include the propagation algorithms discussed in Section 7. In addition, we are extending the OnCheckIn algorithm to consider that norms can regulate the achievement of states. Therefore, we are now working on describing relationships between states and defining the connections between states and actions in order to be able to check for conflicts among norms that regulate the execution of an action and the achievement of a state. We have not considered, in this version, the hierarchy among entities such as roles. If such a relationship was defined, it would have been possible to declare a norm to a more general role and propagate such a norm to more specific roles. As our approach is extensible, to accommodate this relationships is not a difficult task.

## REFERENCES

Cholvy, L. and Cuppens, F. (1995). Solving normative conflicts by merging roles. In *International Conference on Artificial Intelligence and Law*, pages 201–209.

Dung, P. and Sartor, G. (2011). The modular logic of private international law. *Artificial Intelligence and Law*, 19(2-3):233–261.

Elhag, A., Breuker, J., and Brouwer, B. (1999). On the formal analysis of normative conflicts. In *The Twelfth Conference Legal Knowledge Based Systems*, pages 35–46. GNI.

Fenech, S., Pace, G., and Schneider, G. (2009). Automatic conflict detection on contracts. In *Theoretical Aspects*

*of Computing - ICTAC 2009*, volume 5684 of *Lecture Notes in Computer Science*, pages 200–214. Springer.

Figueiredo, K. and Silva, V. (2011). Modeling norms in multi-agent systems with norm-ml. In *Coordination, Organization, Institutions and Norms in Agent Systems IV*, number 6541 in LNAI, pages 39–57. Springer.

Gaertner, D., Garcia-Camino, A., Noriega, P., and Vasconcelos, W. (2007). Distributed norm management in regulated multi-agent systems. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 624–631. ACM.

Garcia-Camino, A., Noriega, P., and Rodrigues-Aguilar, J. (2007). An algorithm for conflict resolution in regulated compound activities. In *Engineering Societies in the Agents World VII*, number 4457 in LNCS, pages 193–208. Springer.

Giannikis, G. and Daskalopulu, A. (2009). Normative conflicts - patterns, detection and resolution. In *International Conference on Web Information Systems and Technologies*, pages 527–532.

Gunay, A. and Yolum, P. (2013). Engineering conflict-free multiagent systems. In *Workshop on Engineering Multi-Agent Systems (EMAS)*, pages 192–207.

Horling, B., Lesser, V., Vincent, R., Wagner, T., Raja, A., Zhang, S., Decker, K., and Garvey, A. (1999). The taems white paper.

Horridge, M. and Bechhofer, S. (2011). The OWL API: A Java API for OWL ontologies. *Semantic Web Journal (Special Issue on Semantic Web Tools and Systems)*, 2(1):11–21.

Hubner, J., Sichman, J., and Boissier, O. (2007). Developing organised multiagent systems using the moise+ model: programming issues at the system and agent levels. *International Journal of Agent Oriented Software Engineering*, 1(3/4):370–395.

Kollingbaum, M., Vasconcelos, W., Garcia-Camino, A., and Norman, T. (2008a). Conflict resolution in norm regulated environments via unification and constraints. In *Declarative Agent Languages and Technologies V*, number 4897 in LNCS, pages 158–174. Springer.

Kollingbaum, M., Vasconcelos, W., Garcia-Camino, A., and Norman, T. (2008b). Managing conflict resolution in norm-regulated environments. In *Engineering Societies in the Agents World VIII*, number 4995 in LNCS, pages 55–71. Springer.

Li, T., Balke, T., De Vos, M., Padget, J., and Satoh, K. (2013a). International conference on legal knowledge and information systems (jurix). In *Legal Conflict Detection in Interacting Legal Systems*.

Li, T., Balke, T., De Vos, M., Satoh, K., and Padget, J. (2013b). Detecting conflicts in legal systems. In *New Frontiers in Artificial Intelligence*, number 7856 in LNCS, pages 174–189. Springer.

Oren, N., Luck, M., Miles, S., and Norman, T. (2008). An argumentation inspired heuristic for resolving normative conflict. In *Coordination, Organisations, Institutions and Norms in Agent Systems*, number 5428 in LNAI. Springer.

Vasconcelos, W., Kollingbaum, M., and Norman, T. (2007). Resolving conflict and inconsistency in norm regulated virtual organizations. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 632–639. ACM.

Vasconcelos, W., Kollingbaum, M., and Norman, T. (2009). Normative conflict resolution in multi-agent systems. *Journal of Autonomous Agents and Multi-Agent Systems*, 19(2):124–152.

Vasconcelos, W. and Norman, T. (2009). Contract formation through preemptive normative conflict resolution. In *International Conference of the Catalan Association for Artificial Intelligence*, pages 179–188. ACM.

W3C (2004). The organization ontology.

Weyns, D., Parunak, H., Michel, F., Holvoet, T., and Ferber, J. (2005). Environments for multiagent systems state-of-the-art and research challenges. In *Environments for Multi-agent Systems*, number 3374 in LNCS, pages 1–47. Springer.

Wooldridge, M. (2000). *Reasoning about Rational Agents*. MIT Press.