A Novel Tool for Detecting Indirect Normative Conflicts in Multi-agent Systems

Jéssica Soares dos Santos[1] and Viviane Torres da Silva[2]
[1]*Computer Science Department, Universidade Federal Fluminense, Niterói, Brazil*
[2]*IBM Research (on leave from Universidade Federal Fluminense), Rio de Janeiro, Brazil*

Keywords: Multi-agent Systems, Norms, Conflict Detection, Ontology, WordNet.

Abstract: Norms are usually applied in Multi-Agent Systems to regulate the behavior of software agents and maintain social order. Those systems can be regulated by multiple norms and require a mechanism to verify whether the set of norms is conflict-free or not. The detection of indirect normative conflicts is not a trivial task since they only can be identified when the detection mechanism is able to infer that different elements that compose two norms are related in some way. In this research, we propose a mechanism to detect normative conflicts by combining two different approaches. The former uses information from a domain ontology that stores relationships that are exclusive of the MAS. The latter uses information from a lexical database called WordNet that stores relationships among concepts of the real world. This research results in the implementation of a tool with a robust mechanism for normative conflict detection that can be used during the design of a MAS.

1 INTRODUCTION

In Multi-agent Systems (MAS), norms are being used in order to control and restrict the software agents behavior. A way of regulation is needed in those systems to avoid the occurrence of undesirable actions since software agents are autonomous entities that can be independently designed. When MAS are governed by multiple norms, it is essential the existence of a mechanism to verify whether such norms contradict each other or not. When there is a contradiction between two norms addressed to the same agent of a MAS we say that there is a normative conflict between the norms. In such a case, the agent cannot comply with both norms simultaneously without violating one of them. There are two kinds of normative conflicts known in the literature, as follows: (i) direct conflicts: they are conflicts that involve norms addressed to the same elements but that have contradictory or opposite modalities, i.e., a prohibition *versus* an obligation or permission regulating the same behavior; and (ii) indirect conflicts: they are conflicts that involve norms that are addressed to different but related elements. Direct conflicts can be easily detected through a direct analysis of the norm elements. On the other hand, it is a challenge to detect indirect conflicts since the conflicting norms are addressed to different elements and the conflict can only be detected when relationships among the norm elements are identified. The detection of indirect conflicts among norms is a topic that is being widely studied in MAS. However, the detection processes of all approaches that we have surveyed in the literature are only possible when the application designer specifies relationships among the elements that compose the norms of the system in a document or an axiom, for instance (see Section 2). A way of detecting indirect conflicts by using the lexical database WordNet (Miller, 1995) was described in (Santos and Silva, 2016). The WordNet stores words and semantic relationships among them, for instance, the words “start” and “stop” are verbs that are related in WordNet because they denote opposite actions in the real world. In this example, such a relationship can be used to detect indirect conflicts between two norms where one obliges the agent *to start* and the other obliges the same agent *to stop*. Using the WordNet, the application designer does not need to worry about specifying relationships that are not specific of the MAS (domain-independent relationships). Additionally, another advantage is that norms of a MAS can be defined by different designers and WordNet can be useful to unify the syntax of the norms, as synonymous words are stored together in WordNet. However, a MAS may need to consider specific relationships, i.e., relationships that do not exist in the real world but exist in the MAS. For instance, relations-

Santos, J. and Silva, V.
A Novel Tool for Detecting Indirect Normative Conflicts in Multi-agent Systems.
DOI: 10.5220/0006598700700079
In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018) - Volume 1*, pages 70-79
ISBN: 978-989-758-275-2

hips that specify that a given software agent inhabits a given environment of the MAS. Those relationships cannot be captured only by using the WordNet. For this reason, in this paper, we present a novel mechanism that combines an approach of conflict detection that uses the WordNet with an approach that uses an ontology to identify domain dependent relationships and detect indirect conflicts among norms of a MAS. By using this mechanism, the application designer only needs to specify in a domain ontology relationships that are exclusive of the given MAS. The remainder of this paper is organized as follows: Section 2 presents other approaches that deal with conflicts among norms in MAS. Section 3 presents all background information needed to the understanding of our research. Section 4 describes our tool and the steps performed by the conflict detection mechanism. Section 5 presents a case study in order to demonstrate the execution of our mechanism. In Section 6, we present our conclusions, highlight limitations and point out suggestions for future work.

2 RELATED WORK

There are many approaches in the literature that propose a means to detect conflicts among norms of a MAS. Most of them can detect direct conflicts (Neto; Silva; Lucena, 2012), (Neto; Silva; Lucena, 2013), (Gaertner et al., 2007), (Vasconcelos et al., 2012) and others can also detect some kinds of indirect conflicts (Şensoy, et al., 2012), (Aphale; Norman; Şensoy, 2013), (Kollingbaum et al., 2007), (Vasconcelos; Kollingbaum; Norman, 2009), (Zahn, 2015), (Fenech; Pace; Schneider, 2008), (Giannikis and Daskalopulu, 2011). The strategies of detection of conflicts can be classified according to when the strategy is applied. The detection process can occur at design time or at runtime. Design time strategies (Fenech; Pace; Schneider, 2008), (Şensoy, et al., 2012), (Aphale; Norman; Şensoy, 2013), (Zahn, 2015) detect conflicts during the design/specification phase of the MAS, that is, before the execution of the MAS. In this case, normative conflicts are detected before they occur and the system designer can verify if the set of norms of a MAS is conflict-free. A disadvantage of adopting design time strategies is that some conflicts only can be detected during the execution of the system (conflicts that depend on the execution order of runtime events). On the other hand, runtime strategies (Kollingbaum et al., 2007), (Vasconcelos; Kollingbaum; Norman, 2009), (Giannikis and Daskalopulu, 2011) detect conflicts during the execution of the MAS. Usually, when an approach uses a runtime detection strategy, it also presents a strategy to resolve conflicts. A disadvantage of runtime strategies is that they can be computationally expensive and it can impact on the performance of the MAS.

Most approaches that can detect indirect normative conflicts consider relationships among actions to do their analysis. The researches presented in (Şensoy, et al., 2012), (Aphale; Norman; Şensoy, 2013), (Kollingbaum et al., 2007) and in (Vasconcelos; Kollingbaum; Norman, 2009) analyze the side-effects of the performance of the actions in order to detect indirect conflicts. The work in (Vasconcelos; Kollingbaum; Norman, 2009) and the approach described in (Zahn, 2015) verify if actions are related by a composition relationship in order to detect conflicts. The researches in (Fenech; Pace; Schneider, 2008), (Giannikis and Daskalopulu, 2011) and (Zahn, 2015) take into account a relationship of orthogonality between actions, which relate actions that cannot be performed at same time. The work in (Zahn, 2015) also considers hierarchy among entities, and relationships that relate an entity to the environment that it inhabits, or an entity to the role it plays, for instance.

Our research differs from the others because it can detect indirect conflicts that occur due to relationships that were not defined by the application designer, and in addition of considering all relationships among actions, entities and contexts defined by (Zahn, 2015), we also consider other relationships that are not considered by other approaches, such as, synonymy and antonymy as detailed in Section 3.

We present a deeper discussion about techniques of detection and resolution of normative conflicts in MAS in a previous work (Santos et al., 2017).

3 BACKGROUND

In this section, we present the essential aspects of our proposal. First, we present the norm definition adopted. The norm definition is an important factor, since all kinds of conflicts that can be detected depend on the norm expressivity, i.e., the elements that a norm can represent. After that, we list the relationships that the mechanism to detect conflicts will investigate between the elements of the norms. Our research combines two different approaches of conflict detection whose relationships are listed separately.

3.1 Norm Definition

We assume that a norm obliges, permits or prohibits an entity to perform an action that can be applied to a specific object. For instance, a norm can say that

an agent is obliged *to drive* a *car*, where *drive* is the action and *car* is the object. The object is an optional element of our norm definition, i.e., a norm can regulate an action that is not associated with an object.

Definition: A norm is a tuple in the form

$$n = \langle id, deoC, c, e, act(obj), ac, dc \rangle$$

where *id* is the norm identifier; *deoC* is the deontic concept that determines the modality of the norm $deoC \in \{obligation, permission, prohibition\}$; $c \in C$ is the context where the norm is defined (it can be an organization $o \in O$ or an environment $env \in Env$); $e \in E$ is the entity being regulated by the norm. An entity e may be an agent $a \in A$, an organization $o \in Org$ or a role $r \in R$; $act \in Act$ is the action being regulated; $obj \in Obj$ is the object associated with the action. The object is an optional field; and $ac \in Cd$ and $dc \in Cd$ are dates that, respectively, activate and deactivate the norm. The symbol "_" can be used to determine that a norm regulates all entities of a specific context.

3.2 Domain Ontology Relationships

The mechanism proposed to detect conflicts is able to receive an ontology as input, specifying the relationships of the application domain. The relationships that can be specified in the ontology by the application designer are listed below, as follows:

Inhabit: it relates an entity to the environment that it inhabits. This relationship indicates that if a norm regulates an environment, the norm also regulates the entities that inhabit such an environment.

Play: it relates an entity to the roles it can assume. This relationship indicates that if a norm regulates a role, the norm also regulates the entities that play such a role.

Ownership: it defines the roles that belong to a given organization. This relationship indicates that if a norm regulates an organization, the norm also regulates the roles that belong to the organization.

Hierarchy: it defines that an element is super element of another one. This relationship indicates that if there is a norm regulating a super context/entity, the norm also regulates their sub contexts/entities.

Refinement: it defines that an action is the specialization of another one.

Composition: it defines that an action (called *whole action*) is composed of other actions (called *part actions*).

Orthogonality: it defines actions that cannot be performed at the same time by the same entity or related entities.

Dependency: it determines that an action (called *client action*) is a precondition to the performance of another one (called *dependent action*).

3.3 WordNet Relationships

Our approach uses the WordNet database to find relationships between contexts, entities, actions and objects (associated with actions).

Synonymy: words that denote the same concept are grouped in a same set (called synset) in the WordNet and are related by the relationship *Synonymy*. We map the contexts described in the norms to nouns and verify if they are related through the relationship *Synonymy*. For instance, if there is a norm whose context is *United States of America* and other one whose context is *USA*, the algorithm will conclude that both contexts are equivalents and that both norms are applied to the same context. Similarly, by using the relationship *Synonymy* we can infer that two norms that are, in principle, addressed to different entities, in fact, refer to the same entity. For instance, *physician* and *doctor* are nouns that denote a *licensed medical practitioner*. We also use the relationship *Synonymy* to map the actions to verbs and objects to nouns, and identify that the actions/objects of two norms are equivalent. For instance, the actions *to collaborate* and *to cooperate* are related by the relationship *Synonymy*.

Specialization: it is described in the WordNet as "*Hyponymy/Hypernymy*" and relate a noun that denotes an element to its respective sub-elements contexts and super-contexts. For instance, the context *hospital* is a sub context of *medical institution*, since a *hospital* is a *medical institution*. Similarly, this relationship can be used to detect that a sub-entity is related to a super-entity. When a norm is applied to a super-entity, such a norm is propagated to its sub-entities. For instance, if a norm is applied to the role *doctor*, supposing that there is the role *angiologist* in the domain, the norm also is applied to all entities that are playing the role *angiologist*. We also use this relationship to detect relationships among objects. For instance, the object *train* is a specialization of the object *public transport*, because a *train* is a kind of *public transport*. The relationship *Specialization* among verbs is defined as "*Troponymy/Hypernymy*" in the WordNet and is used to identify that a sub-action is related to a super-action. For instance, in the WordNet the verbs *to move* and *to walk* are related by this relationship since *to walk* is a way of *to move*.

Part-Whole: it is described in the WordNet as "*Meronymy/Holonymy*" and relates an element that denotes a part to an element that denotes a whole. Usually, this relationship is used to relate geographic areas in the WordNet. For instance, the context *USA* is part of the context *North America*. Similarly, the *intensive care unit* is part of the *hospital* and *sacristy* is a part of *church*. This relationship can be app-

lied to detect related objects, for instance, the objects *window* and *car* are related by the relationship *Part-Whole* because a *window* is part of a *car*.

Entailment: it relates an action that entails another one. For instance, when someone *buy* something it must *pay* for it. Then, the verbs *to buy* and *to pay* are related in the WordNet by the relationship *Entailment*.

Antonymy: it relates an action to its opposite. For instance, *to move* and *to stop* are verbs related in the WordNet by the relationship *Antonymy* because they denote opposite actions.

4 CONFLICT CHECKER TOOL

In this section, we describe how our proposal of conflict checker[1] was implemented. Our tool was developed using Java programming language and NetBeans IDE 8.0 integrated with the OWL-API (Horridge and Bechhofer, 2011), which is the API that we use to read an OWL ontology. The tool uses the JWNL library (Walenz and Didion, 2011) to perform an offline searching in the WordNet. Our algorithm receives as input a domain ontology that describes domain elements (contexts, entities, actions, objects), the set of norms and, optionally, domain-dependent relationships. After that, it combines two approaches of conflict detection and is divided into two steps, as follows:

1. Detection of domain-dependent conflicts by using relationships described in the domain ontology (*Domain Conflict Checker*);
2. Detection of domain-independent conflicts by using relationships described in the WordNet (*WordNet Conflict Checker*).

Both approaches are divided into the following sub-steps:

a. Propagation of norms according to the relationships;
b. Grouping norms in sets according to their similarity;
c. Verification of time intersection between each pair of norms belonging to the same set;
d. Application of rules to identify conflict patterns;

After receiving the domain ontology describing the relationships, entities, contexts, actions, objects and the norms considered by the MAS, the algorithm performs a propagation of norms. During propagation, norms addressed to general entities and contexts

[1] goo.gl/7TmYPX

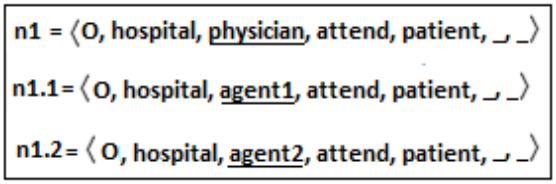

Figure 1: Example of norm propagation.

are addressed to specific entities and contexts of the domain (sub-step a). For instance, let us consider that there are two agents called *agent1* and *agent2* that are playing the role *physician* in the MAS (relationship *play*). Then, if the application designer specifies a norm *n1* that is addressed to the role *physician*, the propagation process will create two new norms *n1.1* and *n1.2* that will be composed of the same elements of *n1* but will be addressed to the agents *agent1* and *agent2* (see Figure 1). However, the propagation of norms according to domain-dependent relationships can generate inconsistent norms if it contradicts other relationships defined in the domain ontology. When it occurs, such norms must be discarded. The rules to discard inconsistent norms are described in Table 1. Since the propagation of contexts and entities can generate multiple norms, to reduce the number of comparisons needed, norms that have the same entity and context are grouped in the same group (sub-step b). For instance, suppose that exists a set of norms: *n3*, *n4*, *n5*, *n6*, *n7* and *n8*. The norms *n3*, *n6* and *n8* are associated with the context *Brazil* and regulate the entity *agent3*; the norm *n5* is associated with the context *Argentina* and regulates the entity *agent4*; and the norms *n4* and *n7* are addressed to the *agent5* and to the context *USA* and *United States of America*, respectively. Note that the contexts *USA* and *United States of America* are synonyms. Then, in this example, the sub-step of grouping norms will create three sets of norms (see Figure 2) and only norms belonging to a same set will be compared in the next sub-step. Only norms addressed to the same (or equivalent) entities and applied to the same (or equivalent) contexts

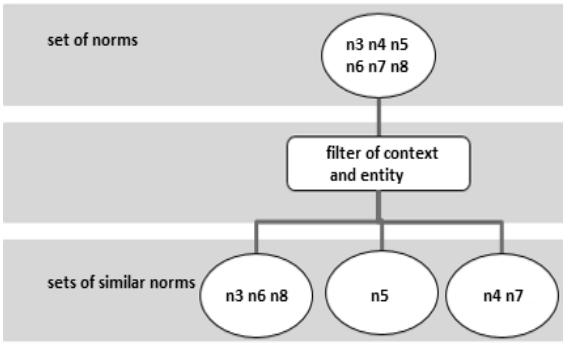

Figure 2: Example of grouping norms according to their similarity.

Table 1: Rules to discard inconsistent norms generated by the propagation process.

Kind of propagation	**Discard rule**
Hierarchy among contexts that are environments	The entity of the original norm is an organization/agent and is not defined in the domain ontology that such entity inhabits the environment of the propagated norm (*inhabit*)
Hierarchy among contexts that are organizations	(i) the entity of the original norm is an organization that is not a suborganization (*hierarchy*) of the context of the propagated norm and is not the context of the propagated norm (ii) the entity of the original norm is a role that is not related to the organization that is the context of the propagated norm (*ownership*)
Inhabit between contexts that are environments and contexts that are organizations	(i) the entity of the original norm is an organization that is not a suborganization (*hierarchy*) of the context of the propagated norm and is not equal to the context of the propagated norm (ii) the entity of the original norm is a role that is not related (*ownership*) to the context of the propagated norm (that is an organization)
Inhabit between contexts that are environments and entities that are organizations	The entity of the propagated norm is not a suborganization (*hierarchy*) of the entity of the original norm

can conflict. Next, for each pair of norms of a same group, the algorithm verifies if there is an intersection between the activation and deactivation conditions of the given two norms (sub-step c) and, if so, the algorithm analyzes the actions, objects and the deontic concepts of the norms in order to verify if the pair of norms is conflicting (sub-step d). The norms that were propagated and the conflicts detected are passed to the second step as an input parameter. The algorithm performs norm propagation considering relationships described in the WordNet (sub-step a). Norms are grouped together in sets when they are addressed to the same or to a synonym entities and contexts (sub-step b). The algorithm verifies if there is a time intersection to each pair of norms of the same set (sub-step c). The algorithm applies the conflict rules to detect conflicting patterns considering the relationships of WordNet (sub-step d). To conclude that two norms are in conflict, the actions defined in the norms must be analyzed with their objects (when the action involves objects). The objects of norms are mapped to WordNet nouns and their relationships are analyzed. Finally, the algorithm exhibits all conflicts detected and the domain-dependent and domain-independent relationships that were identified. Note that in Section 5, we exhibit some conflicting patterns, that is, patterns that indicate that two norms cannot be adopted at the same time and for this reason are in conflict. The complete list of conflicting patterns that have been defined involving domain-independent/domain-dependent relationships are described in (Santos and Silva, 2016) and (Zahn, 2015), respectively.

Figure 3 illustrates the graphical interface of the conflict checker tool. The button “Upload” is used to select an OWL ontology. The button “Execute” performs the detection process. The field “Ontology Description” exhibits the information described in the ontology received as input, that is, norms, contexts, entities, actions, objects and relationships of the domain application described by the application designer. The field “Domain Conflict Checker” exhibits the propagations due to domain-dependent relationships, the pairs of norms compared and indicates whether each pair is conflicting or not and the reason. Similarly, the field “WordNet Conflict Checker” exhibits the propagations due to domain-independent relationships, exhibits the pairs of norms compared and indicates the reason of conflict. The field “Conclusion” groups information from the fields “Domain Conflict Checker” and “WordNet Conflict Checker” and exhibits all conflicting patterns found to each pair of norms compared. The “Reset” button cleans all information exhibited on the screen to be able to select a new ontology and perform another detection process.

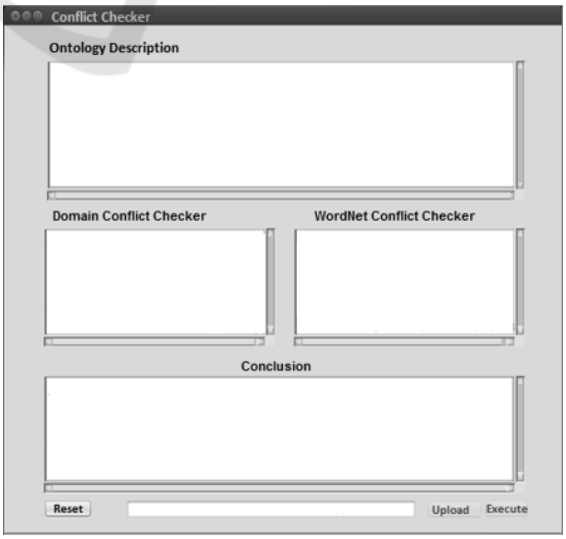

Figure 3: GUI of the Conflict Checker.

5 CASE STUDY

In this section, we present a case study to illustrate the process of conflict detection. Our tool will use norms of an e-commerce scenario.

5.1 Definition of the Case Study

The type of e-commerce contract of this case study is an agreement involving two parties called contractor and contracted, where the contracted provides a service of management and availability of a virtual shop and the contractor is the entity that has interest in acquiring such a service, that is, it has an interest in making its products available for sale on the Internet. The selected contract is a default agreement that defines the basic conditions for the establishment of an agreement between the contracted and any contractor, where both inhabit the context of USA. In the following case study, the deontic concepts "obligation", "permission" and "prohibition" are represented as "O", "P", and "F", respectively. Since the selected contract has many norms, we have selected only a few to demonstrate the process of conflict detection in a simplified way, as follows:

1) The contractor is prohibited from being unaware of any part of the contract
$\langle N1, F, USA, contractor, unaware(contract), \_, \_\rangle$

2) The contractor is obliged to agree to the contract
$\langle N2, O, USA, contractor, agree(contract), \_, \_\rangle$

3) The contractor cannot reproduce the tools provided by the contracted. Such tools are software programs of the virtual shop management
$\langle N3, F, USA, contractor, reproduce(tool), \_, \_\rangle$

4) The contractor also cannot market the tools made available by the contracted
$\langle N4, F, USA, contractor, market(tool), \_, \_\rangle$

5) The contractor cannot send e-mails to users of the virtual shop
$\langle N5, F, USA, contractor, send(email), \_, \_\rangle$

6) The contractor must purchase the software to be installed in the virtual shop environment
$\langle N6, O, USA, contractor, purchase(software), \_, \_\rangle$

7) The contractor is obliged to deliver the products sold in the website
$\langle N7, O, USA, contractor, deliver(product), \_, \_\rangle$

8) The contractor is prohibited to access servers of the contracted
$\langle N8, F, USA, contractor, access(server), \_, \_\rangle$

9) The contracted may exclude products from the virtual shop, for instance, when it considers that the sale of them is improper
$\langle N9, P, USA, contracted, exclude(product), \_, \_\rangle$

10) The contracted may change the contract
$\langle N10, P, USA, contracted, change(contract), \_, \_\rangle$

11) The contractor is obliged to include a freight to the sales
$\langle N11, O, USA, contractor, include(freight), \_, \_\rangle$

In order to demonstrate the operation of our proposal for the detection of conflicts, we will consider that there is a contractor party called *Company_1* that has an interest in using the services of the contracted party and that also has preconditions in relation to the aforementioned agreement with the contracted party. Therefore, our conflict resolution consists in analyzing the norms of the default contract defined by the contracted party together with the specific norms of a given contractor in order to identify inconsistences. Let us assume that *Company_1* has the following norms in relation to the default contract established by the contracted:

12) *Company_1* may be unaware of clauses of the contract
$\langle N12, P, USA, Company\_1, unaware(clause), \_, \_\rangle$

13) *Company_1* may disagree with the contract clauses
$\langle N13, P, USA, Company\_1, disagree(clause), \_, \_\rangle$

14) *Company_1* may copy tools from the contracted
$\langle N14, P, USA, Company\_1, copy(tool), \_, \_\rangle$

15) *Company_1* may login on computers from contracted
$\langle N15, P, USA, Company\_1, login(computer), \_, \_\rangle$

16) *Company_1* may exclude the freight of purchases made in California during the month of December 2017
$\langle N16, P, California, Company\_1, exclude(freight),$ 12/01/2017 00:00:00, 12/31/2017 00:00:00$\rangle$

17) The contracted cannot change the contract
$\langle N17, F, USA, contracted, change(contract), \_, \_\rangle$

18) *Company_1* may send spam, for instance, to promote the sale of products or other services
$\langle N18, F, USA, Company\_1, send(spam), \_, \_\rangle$

19) The contracted cannot perform actions associated with the management of products from the virtual shop, that is, it cannot insert, edit or exclude products
$\langle N19, F, USA, contracted, manage(product), \_, \_\rangle$

20) *Company_1* cannot pay for any software that will be installed in the virtual shop
$\langle N20, F, USA, Company\_1, pay(software), \_, \_\rangle$

The application designer only needs to explicitly describe in the ontology the existing relationships of the application domain. Thus, in this case it was only necessary to explicitly define the following relationships within the domain ontology: (i) *Play* is declared

between *Company_1* and *contractor*, indicating that *Company_1* is an entity that plays the role of contractor; and (ii) *Refinement* is declared between the action *manage* and the actions *register*, *edit* and *delete*.

5.2 Conflicts Detected

Conflicts are detected by combining the information provided by the domain ontology (described by the application designer) with the information from WordNet database. Initially the domain-dependent propagation occurs, considering relationships defined in the ontology. This step will create the following norms:

Domain dependent propagation - Propagation between entities:
(Contractor - *Company_1*)

$\langle N1.1, F, B, Company\_1, unaware(contract), \_, \_\rangle$
$\langle N2.3, O, USA, Company\_1, agree(contract), \_, \_\rangle$
$\langle N3.4, F, USA, Company\_1, reproduce(tool), \_, \_\rangle$
$\langle N4.5, F, USA, Company\_1, commercialize(tool), \_, \_\rangle$
$\langle N5.6, F, USA, Company\_1, send(email), \_, \_\rangle$
$\langle N6.7, O, USA, Company\_1, purchase(software), \_, \_\rangle$
$\langle N7.8, O, USA, Company\_1, deliver\ (product), \_, \_\ \rangle$
$\langle N8.9, F, USA, Company\_1, access(server), \_, \_\rangle$
$\langle N11.2, O, USA, Company\_1, include\ (freight), \_, \_\rangle$

After propagation and domain-dependent processing, the original norms are joined to the propagated norms and independent domain processing is performed. By using WorNet, the mechanism detects that one of the contexts of the domain (California) is part of other context of the domain (USA). Then, all norms related to USA are always related to California. This norm propagation which will result in the following norms:

Domain-independent propagation - Propagation between contexts:
(USA - California)

$\langle N1.1, F, California, contractor, unaware(contract), \_, \_\rangle$
$\langle N10.2, P, California, contracted, change(contract), \_, \_\rangle$
$\langle N11.3, O, California, contractor, include(freight), \_, \_\rangle$
$\langle N12.4, P, California, Company\_1, unaware(clause), \_, \_\rangle$
$\langle N13.5, P, California, Company\_1, disagree(clause), \_, \_\rangle$
$\langle N14.6, P, California, Company\_1, copy(tool), \_, \_\rangle$
$\langle N15.7, P, California, Company\_1, login(computer), \_, \_\rangle$
$\langle N17.8, F, California, contracted, edit(contract), \_, \_\rangle$
$\langle N18.9, P, California, Company\_1, send(spam), \_, \_\rangle$
$\langle N19.10, P, California, contracted, exclude(product), \_, \_\rangle$
$\langle N2.11, O, California, contractor, agree(contract), \_, \_\rangle$
$\langle N20.12, F, California, Company\_1, pay(software), \_, \_\rangle$
$\langle N3.13, F, California, contractor, reproduce(tool), \_, \_\rangle$
$\langle N4.14, F, California, contractor, commercialize(tool), \_, \_\rangle$
$\langle N5.15, F, California, contractor, send(email), \_, \_\rangle$
$\langle N6.16, O, California, contractor, purchase(software), \_, \_\rangle$

Table 2: Conflicts detected in the case study presented.

Conflict	Conflict Pattern	Norms
(N1, N12)	-Time intersection -Same Action -Object *Part-Whole* (clause, contract) - F x P	(⟨N1.1.20, F,California, Company_1, unaware (contract), _, _⟩, ⟨N12.4, P, California, Company_1, unaware (clause),_, _⟩)
(N2, N13)	-Time intersection -Action *Antonymy* (agree, disagree) -Object *Part-Whole* (clause, contract) -O x P	(⟨N2.3,O,USA,Company_1, agree (contract), _, _⟩, ⟨N13,P,USA,Company_1, disagree (clause), _, _⟩) (⟨N2.3.22,O,California, Company_1,agree(contract), _,_⟩, ⟨N13.5,P,California, Company_1,disagree (clause),_,_⟩)
(N3, N14)	-Time intersection -Action *Synonymy* (copy, reproduce) -Same object -F x P	(⟨N3.4,F,USA,Company_1, reproduce(tool),_,_⟩, ⟨N14,P,USA,Company_1, copy (tool),_,_⟩) (⟨N3.4.23,F,California, Company_1,reproduce (tool),_,_⟩, ⟨N14.6,P,California, Company_1,copy(tool),_, _⟩)
(N5, N18)	-Time intersection -Same action -Object *Specialization* (spam, email) -F x P	(⟨N5.6,F,USA,Company_1, send(email),_,_⟩, ⟨N18,P,USA,Company_1, send(spam),_,_⟩) (⟨N5.6.25,F,California, Company_1,send(email),_,_⟩, ⟨N18.9,P,California, Company_1,send(spam),_,_⟩)
(N6, N20)	-Time intersection -Action *Entailment* (purchase, pay) -Same object -O x F	(⟨N6.7,O,USA,Company_1, purchase(software),_,_⟩, ⟨N20,F,USA,Company_1, pay(software),_,_⟩) (⟨N6.7.26,O,California, Company_1,purchase (software),_,_⟩, ⟨N20.12,F,California, Company_1,pay (software)_,_⟩)
(N9, N19)	-Time intersection -Action *Refinement* (exclude, manage) -Same object -F x P	(⟨N9,F,USA,contracted, manage(product),_,_⟩, ⟨N19,P,USA,contracted, exclude(product),_,_⟩)
(N11, N16)	-Time intersection -Action *Antonymy* (exclude, include) -Same object -O x P	(⟨N11.2.21,O,California, Company_1,include (freight),_,_⟩, ⟨N16,P,California, Company_1,exclude(freight), 01/12/2016 00:00:00, 12/31/2016 00:00:00⟩)

$\langle N7.17, O, California, contractor, deliver(product), \_, \_\rangle$
$\langle N8.18, F, California, contractor, access(server), \_, \_\rangle$
$\langle N9.19, F, California, contracted, manage(product), \_, \_\rangle$
$\langle N1.1.20, F, California, Company\_1, unaware(contract), \_, \_\rangle$
$\langle N11.2.21, O, California, Company\_1, include(freight), \_, \_\rangle$
$\langle N2.3.22, O, California, Company\_1, agree(contract), \_, \_\rangle$
$\langle N3.4.23, F, California, Company\_1, reproduce(tool), \_, \_\rangle$
$\langle N4.5.24, F, California, Company\_1, commercialize(tool), \_, \_\rangle$
$\langle N5.6.25, F, California, Company\_1, send(email), \_, \_\rangle$

⟨*N6.7.26,O,California,Company_1,purchase(software),_,_*⟩
⟨*N7.8.27,O,California,Company_1,deliver(product),_,_*⟩
⟨*N8.9.28,F,California,Company_1,access(server),_,_*⟩

Note that norms whose *id* is a triple numbering (for instance, *N5.6.25*) are norms that were propagated two times (domain-dependent propagation and domain-independent propagation).

After the complete analysis, a total of 7 normative conflicts were detected. It is important to emphasize that conflicts can only occur between the norms that are addressed to the same entities and contexts. Such conflicts are listed in Table 2.

Note that the detection mechanism of our tool combines two approaches, then, the conflict between norms *N9* and *N19* was detected based on a domain dependent relationship (relationship *Refinement* between actions). The remaining conflicts were detected based on WordNet relationships involving actions and objects (*Synonymy*, *Antonymy*, *Entailment*, *Specialization*, *Part-Whole*), and based on the relationship *Play* specified on the domain ontology that determines that *Company_1* plays the role of *contractor*.

6 CONCLUSIONS

The detection of conflicts among norms of a MAS is a very important topic that has been studied in MAS. When a system is governed by a large and/or complex set of norms, a process of verification/revision of norms is needed to avoid the occurrence of normative conflicts. However, it is difficult for a human to detect all normative conflicts that may arise in a big set of norms, besides being a task that demands a lot of time and that is prone to errors. In this context, our research aims to automate the conflict detection process, providing a mechanism able to detect direct and indirect normative conflicts. Our research provides means to detect indirect normative conflicts that do not depend on the domain of the application, that is, it is able to detect conflicts even when the system designer does not previously specify the relationships among the elements of the norms, performing a semantic mapping. In addition, we consider cases of conflicts that may occur due to relationships that have not yet been considered in any proposal of other authors, such as: synonyms and antonyms. However, we know that a MAS can be designed to meet the needs of a more specific universe, and therefore we cannot just consider relationships that exist in the real world in our analysis. For this reason, we have combined our approach with the research presented in (Zahn, 2015) and have developed a robust mechanism that can also analyze relationships previously defined in a domain ontology. In short, our proposal to detect normative conflicts was divided into three steps, as follows:

(i) the first step consists in creating a mechanism that maps the elements that compose the norms to words and after that, search for relationships between words. In this step, several cases of normative conflicts were defined that can be inferred using WordNet as source of information. This step is responsible for detecting domain-independent conflicts;

(ii) the second step extends the approach presented in (Zahn, 2015) so that it can be integrated with the mechanism created in the first step in order to detect domain-dependent conflicts;

(iii) the third step consists of integrating the first and second steps and creating a tool for checking indirect conflicts involving relationships that depend or not of the domain.

6.1 Limitations

Among the limitations of our research we can highlight:

(i) the norms of the MAS must be described in an OWL ontology following a specific format. We consider this a limitation of our approach because if the system designer has a set of norms described in natural language he will need to convert this set of norms to the specific format defined in the ontology;

(ii) domain-dependent relationships also need to be described in an ontology and follow the format detailed in (Zahn, 2015);

(iii) the developed tool is not able to distinguish words that are homonymous, that is, words whose spelling is the same, but have different meanings when inserted in different contexts. For example, the word *doctor* may refer to *a person graduated in medicine* (medical context) or *a person who holds a doctorate degree* (academic context). This problem is known as Lexical Disambiguation of Meaning (LDM) or Word Sense Disambiguation (WSD) in Artificial Intelligence (AI) (Ide and Véronis, 1998). Although it is possible to analyze all the elements of a norm to try to infer the meaning of a word in a given context, this step is very costly and is not a guarantee of certainty. In practice, such ambiguities are unlikely to occur during conflict detection because, in general, norms are related to the same domain or related ones. In addition, WordNet can store a

word in different synsets that have a small difference of meaning, which may make the process of disambiguation unnecessary in many cases. An example of this is the verb *kill* that belongs to different synsets whose meanings are very similar, as listed in Table 3.

Table 3: Meanings of the verb kill in WordNet.

	Gloss	Example
(verb) kill	causing death intentionally	"This man killed several people when he tried to rob a bank"
	be fatal	"cigarettes kill"
	deprive of life	"AIDS has killed thousands in Africa"
	causing death unintentionally	"She was killed in the collision of three cars"

Despite these limitations, we consider that our research provided a great contribution to the area of detection of normative conflicts because it has resulted in the creation of a tool that can help a software engineer to design/include norms in a MAS in a consistent way and mainly because it is possible to use such a tool to detect normative conflicts in MAS that do not have previously specified domain relationships. Thus, the software engineer/system designer only needs to specify in a domain ontology relationships that do not occur in the real world.

6.2 Future Work

As suggestions for future work, we point out the following extensions for this research:

(i) to define cases of domain-independent conflicts between norms that regulate states. This can be implemented by using the majority of relationships between defined actions, but by referring to another grammar class. To search for relationships between actions we map actions to verbs and to search for relationships between states we could map states to adjectives;

(ii) to implement the lexical disambiguation process. This process should consider the grammatical class of the word and perform an analysis involving all the elements that compose the norm (context, entity, action, object). The disambiguation process can also investigate the semantic similarity between two words, that is, can use metrics to calculate numerical values that determine the proximity between a word and a concept that contains a certain word. Other information that may be useful for lexical disambiguation are the words contained in the description (called "gloss") of each WordNet synset. In addition, if the WordNet synset is associated with sentences that exemplify the use of its words, such sentences can also be used. However, since most existing LDM methods require high computational complexity, many approaches adopt heuristics of disambiguation, such as (Mihalcea, 2006):

a. the most common sense: words are disambiguated according to the most common sense of the language. In this case, the algorithm should select the first synset that contains the word to be disambiguated (because the WordNet sorts the synsets according to their frequency of use);

b. a sense by discourse: after determining the meaning of a word p, all other occurrences of that word will be attributed to the same meaning;

(iii) to receive as input of the algorithm norms in natural language. In this case, it is necessary to develop a mechanism for preprocessing a text file that contain the norms in order to map them to the norm definition presented in Section 3.1. Note that when the norms are described in natural language the modality of the norm (prohibition, permission, obligation) may not be explicitly described. To identify the modality of the norm, the method can verify, for instance, if the norm has some of the modal verbs, as follows:

a. verbs that denote obligations: *must*, *need*, *ought*, *have to*, *will*;

b. verbs that denote permissions: *can*, *may*, *could*;

c. verbs that denote prohibitions (usually verbs denoting permissions and obligations accompanied by a term denoting denial): *cannot*, *should not*, *must not*, *do not have*.

REFERENCES

Aphale, M., Norman, T. J., and Şensoy, M., 2013. Goal-directed policy conflict detection and prioritisation. In Aldewereld, H., Sichman, J.S., (Eds), Coordination, organisations, institutions and norms in agent systems VIII, volume 7756 of Lecture notes in computer science (pp. 87104). Springer.

Şensoy, M., Norman, T. J., Vasconcelos, W. W., and Sycara, K., 2012. OWL-POLAR: A framework for semantic policy representation and reasoning. Web Semantics: Science, Services and Agents on the World Wide Web, 1213, 148160.

Fenech, S., Pace, G. J., and Schneider, G., 2008. Detection of conflicts in electronic contracts. NWPT 2008, 34.

Gaertner, D., Garcia-Camino, A., Noriega, P., Rodriguez-Aguilar, J. A., and Vasconcelos, W. W., 2007. Distributed norm management in regulated multiagent systems. In Proceedings of the 6th international joint

conference on autonomous agents and multiagent systems, AAMAS 07 (pp. 90:190:8). New York, NY: ACM.

Giannikis, G. K. and Daskalopulu, A., 2011. Normative conflicts in electronic contracts. Electronic Commerce Research and Applications, 10(2), 247267.

Horridge M. and Bechhofer S., 2011. The OWL api: A java api for owl ontologies. Semantic Web, v. 2, n. 1, p. 11-21.

Ide, N. and Véronis, J., 1998. Introduction to the special issue on word sense disambiguation: the state of the art. Computational linguistics, v. 24, n. 1, p. 2-40.

Kollingbaum, M. J., Norman, T. J., Preece, A., and Sleeman, D., 2007. Norm conflicts and inconsistencies in virtual organisations. In P. Noriega, J. Vázquez-Salceda, G. Boella, O. Boissier, V. Dignum, N. Fornara, & E. Matson (Eds.), Coordination, organizations, institutions, and norms in agent systems II, volume 4386 of Lecture notes in computer science (pp. 245258). Berlin: Springer.

Mihalcea, R., 2006. Knowledge-based methods for WSD. Word Sense Disambiguation: Algorithms and Applications, pp. 107-131.

Miller, G. A., 1995. WordNet: a lexical database for English, Communications of the ACM, 38(11), 3941.

Neto, B. F. S., Silva, V. T., and Lucena, C. J. P., 2012. An architectural model for autonomous normative agents. In L. Barros, M. Finger, A. Pozo, G. Gimenénez-Lugo, & M. Castilho (Eds.), Advances in artificial intelligenceSBIA 2012, Lecture notes in computer science (pp. 152161). Berlin: Springer.

Neto, B. F. S., Silva, V. T., and Lucena, C. J. P., 2013. Developing goal-oriented normative agents: The NBDI architecture. In J. Filipe, & A. Fred (Eds.), Agents and artificial intelligence, volume 271 of Communications in computer and information science (pp. 176191). Berlin: Springer.

Santos, J. S., and Silva, V. T., 2016. Identifying Domain-Independent Normative Indirect Conflicts. In: 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), 2016, San Jose. p. 536.

Santos, J. S., Zahn, J. O., Silvestre, E. A., Silva, V. T., and Vasconcelos, W. W., 2017. Detection and resolution of normative conflicts in multi-agent systems: a literature survey. Autonomous Agents and Multi-Agent Systems, pp. 1-47. DOI: [ONLINE] DOI: 10.1007/S10458-017-9362-Z.

Vasconcelos, W. W., Kollingbaum, M. J., and Norman,T. J., 2009. Normative conflict resolution in multiagent systems. Autonomous Agents and Multi-Agent Systems, 19(2), 124152.

Vasconcelos, W. W., García-Camino, A., Gaertner, D., Rodríguez-Aguilar, J. A., and Noriega, P., 2012. Distributed norm management for multi-agent systems. Expert Systems with Applications, 39(5), 5990 5999.

Walenz, B. and Didion, J., 2011. “JWNL: Java WordNet library.” 2008-05-14]. http://jwordnet. sourceforge, net.

Zahn, J. O., 2015. Um Mecanismo de Verificação de Conflitos Normativos Indiretos. Masters thesis, Instituto de Computação - Universidade Federal Fluminense (IC/UFF), Niterói, Brasil.