Norms in Multiagent Systems: some Implementation Guidelines

Javier Vázquez-Salceda, Huib Aldewereld, and Frank Dignum

Institute of Information and Computing Sciences
Utrecht University, The Netherlands
{javier, huib, dignum}@cs.uu.nl

Abstract. Norms are commonly used in MAS to formally express the expected behaviour of agents in open environments. Current norm formalisms focus on the *declarative* nature of norms. However, in order to be implemented, norms should be translated into *operational* representations. In this paper we continue our work on the implementation of norms by discussing issues on norm enforcement, verifiability and defeasibility. We propose some implementation guidelines, including some mechanisms to be added in agent platforms in order to ease norm implementation.

1 Introduction

In open societies, where heterogeneous agents might deviate from expected behaviour, mechanisms are needed in order to systematize, defend and recommend right and wrong behaviour, along with safe environments to support those mechanisms, thereby inspiring trust into the agents that will join the society. [1] One example of such safe environments is an Electronic Institution [10] [12], where the expected behaviour of agents is described by means of an explicit specification of norms. Such norm specification should be a) expressive enough, b) readable by agents, and c) easy to maintain.

Current work on normative systems' formalization (mainly focused in Deontic-like formalisms [16]) is declarative in nature, focused on the expressiveness of the norms, the definition of formal semantics and the verification of consistency of a given set. In previous work [9] [11] we have focused on the formal definition of norms by means of some variations of deontic logic that includes conditional and temporal aspects [5] [8], and we provided formal semantics. Although the declarative aspects of norms are important, norms should not only have a *declarative* meaning but also an *operational* one in order to be used in MAS. This means that, to be used in practice, norms should be operationally implemented.

It is important to note that implementing norms is not implementing a theorem prover that, using the norm semantics, checks whether a given interaction protocol complies with the norms. The implementation of norms should consider two perspectives: a) the *agent perspective*, analyzing the impact of norms in the agents' reasoning cycle (work on this perspective can be found in [3] [4] [7]), and b) the *institutional perspective*, implementing a safe environment (including the enforcing mechanisms) to ensure trust among parties. As far as we know, the most complete model in literature considering some operational aspects of norms for MAS is the extension of the SMART agent specification framework by López y López, Luck and d'Inverno [17] [18]. The framework aims to represent different kinds of agent societies based on norms. However, no implementation of the architecture applying it to a

[1] Some foundational work in this direction has been done in the ALFEBIITE project [14], in particular in [2].

real problem has been reported in literature, there are no tools to support the development and implementation of a normative multiagent system, and there are no mechanisms defined from the institutional perspective in order to enforce the norms.

In this paper we complement our previous work on norm formalization by focusing on how norms should be operationally implemented in MAS from an institutional perspective (i.e. How to check a norm? How to detect a violation of a norm? How to handle it?). In order to analyze the problem we categorize norms depending on a) whether they are *restrictive* (norms permitting/forbidding actions or situations) or *impositive* (norms forcing an entity to do an action or to reach a state), b) how the start and end of an obligation are detected, and c) the different aspects of the norms to be specified. We also propose a first draft of a machine-readable format for expressing norms, which is not only expressive enough for complex norms (such as those present in *e*Commerce, *e*Government or *e*Care domains) but also useful for implementation in MAS. Our implementation guidelines use the ISLANDER framework for institutions and platform as a starting point.

The paper is organized as follows. In the next section we discuss how normative specification is currently done in the ISLANDER formalism, being the most appropriate for defining institutions. Then, in §3, we discuss the different types of norms one can distinguish and their implementation related issues (including verifiability and violation handling). In §4 we analyze the implementation of combinations of norms where some of the norms become defeasible. We end this paper with our conclusions and outline future lines of research. To illustrate that our approach is quite general and can be used on several domains, we use examples of norms throughout this paper coming from three different domains (electronic auction houses such as Fishmarket, organ and tissue allocation for human transplantation purposes and the access to Dutch police criminal registers).

2 Norms in ISLANDER

The ISLANDER formalism [12] provides a formal framework for institutions [21]. It has proven to be well-suited to model practical applications; ISLANDER has been mainly used in *e*Commerce scenarios, and was used to model and implement an electronic Auction house (the *Fishmarket*).

This formalism views an agent-based institution as a *dialogical system* where all the interactions inside the institution are a composition of multiple dialogic activities (message exchanges). These interactions are structured through agent group meetings called *scenes* that follow well-defined protocols. A second key element of the ISLANDER formalism is the notion of an agent's *role*. Each agent can be associated to one or more roles, and these roles define the scenes the agent can enter and the protocols it should follow. Finally, this formalism defines a graphical notation that not only allows to obtain visual representations of scenes and protocols but is also very helpful while developing the final system, as they can be seen as blueprints.

Furthermore, the AMELI platform [13] allows the execution of electronic institutions, based on the rules provided by ISLANDER specifications, wherein external agents may participate.

2.1 Restrictive Norms in ISLANDER

Most of the norms that can be expressed in ISLANDER are *restrictive norms* in the interaction[2] of agents, enforcing that agents utter only acceptable illocutions according to an intended interaction protocol expressed by means of the performative structure.

The *performative structure* defines the conversations that can take place within the institution and how agents, depending on their role, can move among them. A performative structure can be depicted as:

- a collection of multiple, concurrent *scene* nodes. All constraints in the interaction that apply inside a scene are expressed by the scene protocol. A *scene protocol* is specified by a finite state directed graph where the nodes represent the different states of the conversation and the directed arcs connecting the nodes are labeled with the illocutions that make the scene state evolve.
- *transition* nodes, devoted to mediate different types of connections among scenes and establishing synchronization and parallelism points.
- *labeled arcs*, specifying the role or roles that are able to pass from one to another node in the diagram.

Restrictive norms on the agents' behavour are implicitly represented in the performative structure and scene protocol graphs. Between scenes, an agent cannot pass from one scene to another if there is no explicit path of arcs and transitions in the performative structure connecting both scenes for the role or roles the agent is enacting. Inside the scenes, for each state of the scene protocol graph any illocution that is not explicitly represented as allowed (by means of an outgoing arc) is forbidden. All these restrictions are managed in the AMELI platform by means of the *Governors*, the *Scene Managers* and the *Transition Managers.* In AMELI, external agents do not interact with the e-institution (and other agents in the e-institution) directly, but throught the use of Governors. Governors filter the messages of the agents and exclude illegal and (from an institutional perspective) unknown messages. Norms implemented in the design of the institution are checked and enforced by the Scene and Transition managers that deny access to states and scenes if the role of an agent or its illocutions do not match those specified by the designer of the e-institution.

2.2 Impositive norms in ISLANDER

Apart of the restrictive norms expressed in the performative structure, ISLANDER also allows the introduction of *norms to trigger obligations.* These norms express the consequences of some key actions (e.g. winning an auction) within the institutions. The consequences are expressed as obligations that agents will acquire or satisfy depending on their illocutions within the different scenes (e.g. the obligation to pay the items the agent won). The definition of this kind of norms is composed by:

- *antecedent*: the set of illocutions that, when uttered in a given scene satisfying the given conditions, will trigger the norm, making the set of obligations expressed in the consequent hold.
- *defeasible antecedent*: defines the illocutions that must be uttered in the defined scenes in order to fulfil the obligations

[2] As the ISLANDER formalism views a MAS from a *dialogical perspective*, the only actions that can be modelled and controlled are messages (the *illocutions* [20]).

- *consequent*: a list of obligation expressions.

This kind of norms are handled by the *governors*. Each governor keeps, at any moment, the pending obligations of its associated agent and checks whether agent interactions (the uttered and received illocutions) activate or de-activate the obligations (by checking the antecedent and the defeasible antecedent of the norm).

2.3 Discussion

ISLANDER is a framework which provides a sound model for the domain ontology and has a formal semantics [21]. This is an advantage of its dialogical approach to organizations. However, in ISLANDER the normative aspects are reduced to the afore mentioned protocol (expressed in the performative structure) plus the specification of constraints for scene transition and enactment (the only allowed interactions are those explicitly represented by arcs in scenes), along with the definition of norms that uniquely allow for the firing of obligations. Thus, ISLANDER does not offer expressiveness to specify norms involving prohibitions, permissions, or sanctions. Furthermore, it does not allow the use of temporal operators. And finally, ISLANDER does not allow for the specification of non-dialogical actions.

Our aim is to extend the norms in the ISLANDER formalism with more expressive, abstract norms while providing some mechanisms to implement the enforcement of these norms from the institutional perspective.

3 Norms: types, components and implementation issues

In this section we will focus on indicating possible implementation guidelines related with the different kinds of norms and the components in each of them. There are two main assumptions in our approach. First of all we assume that norms can sometimes be violated by agents in order to keep their autonomy, which can also be functional for the system as a whole as argued in [6]. The violation of norms is handled from the organizational point of view by violation and sanction mechanisms. Secondly we assume that from the institutional perspective the internal state of the external agents is neither observable nor controlable (external agents as black boxes). Therefore, we cannot avoid a forbidden action to be in the goals and intentions of an agent, or impose an obligatory action on an agent to be in their intentions.

In order to implement enforcement mechanisms that are well-found, one has to define some kind of operational semantics first. In general, an operational semantics for norms always comes down to either one of the following:

- Defining constraints on unwanted behaviour.
- Detecting violations and reacting to these violations.

The choice between these two approaches is highly dependent on the amount of control over the addressee of the norms.[3] Prevention of unwanted behaviour can only be achieved if there is full control over the addressee; otherwise, one should define and handle violations (see §3.4).

[3] An analysis of the different kinds of addressee of norms and their impact on implementation is presented in [22].

3.1 Norm Condition Expression Language.

In [22] we characterized norms[4] by whether a) they refer to a state or an action, b) they are conditional, c) they include a deadline, or d) they are norms concerning other norms. Based on this classification, we can specify a generic language for expressing norm conditions. Although this language can be given a formal semantics, we refrain from doing so for now, but refer to [8] [11].

Definition 1 (Norm Condition).

$$\begin{aligned}
\mathsf{NORM\_CONDITION} &:= N(a, S\ \langle \mathsf{IF}\ C\rangle) \mid \mathsf{OBLIGED}(a\ \mathsf{ENFORCE}(N(a, S\ \langle \mathsf{IF}\ C\rangle)))\\
N &:= \mathsf{OBLIGED} \mid \mathsf{PERMITTED} \mid \mathsf{FORBIDDEN}\\
S &:= P \mid \mathsf{DO}\ A \mid P\ \mathsf{TIME}\ D \mid \mathsf{DO}\ A\ \mathsf{TIME}\ D\\
C &:= \mathit{proposition}^{5}\\
P &:= \mathit{proposition}\\
A &:= \mathit{action\ expression}\\
\mathsf{TIME} &:= \mathsf{BEFORE} \mid \mathsf{AFTER}
\end{aligned}$$

Definition 1 shows that norm conditions can either be concerning states, e.g. for a norm such as

$$\mathsf{FORBIDDEN}(\mathit{buyer}, \mathit{account}(\mathit{buyer}, A) \wedge A < 0)$$

or concerning actions, e.g.

$$\mathsf{FORBIDDEN}(\mathit{seller}\ \mathsf{DO}\ \mathit{bid}(\mathit{product}, \mathit{price}))$$

The definition allows the norm condition to be conditional, allowing the expression of norms like

$$\begin{gathered}\mathsf{OBLIGED}((\mathit{user}\ \mathsf{DO}\ \mathit{include}(\mathit{source}(\mathit{Suspect\_data}), \mathit{Criminal\_Register}))\\ \mathsf{IF}\ (\mathit{done}(\mathit{include}(\mathit{Suspect\_data}, \mathit{Criminal\_Register}))))\end{gathered}$$

as well as norm conditions including temporal aspects in the form of deadlines, for instance

$$\begin{gathered}\mathsf{OBLIGED}((\mathit{allocator}\ \mathsf{DO}\ \mathit{assign}(\mathit{heart}, \mathit{recipient}))\\ \mathsf{BEFORE}\ (\mathit{time}(\mathit{done}(\mathit{extraction}(\mathit{heart}, \mathit{donor}))) + 6\mathit{hours}))\end{gathered}$$

The other group of norm conditions that can be expressed in the language defined in definition 1 are those concerning enforcement of norms on other agents.

$$\mathsf{OBLIGED}(\mathit{ONT}\ \mathsf{ENFORCE}(\mathsf{FORBIDDEN}(\mathit{person}\ \mathsf{DO}\ \mathit{sell}(\mathit{organ}))))$$

3.2 Implementation guidelines for norm enforcement

The elements present in the norm expressions (or norm condition) have a direct impact on norm enforcement. For all the norms above, the implementation of enforcement is composed of three related processes:

[4] Note that we will use the term norms wherever Obligations (OBLIGED) as well as Permissions (PERMITTED) or Prohibitions (FORBIDDEN) are meant.

[5] The conditions (C) and propositions (P) are expressed in some kind of propositional logic. This logic can use deontic (cf. [9] [11]), or temporal (cf. [5] [8]) operators. Note however that this logic should at least include some operational operators like, for instance, DONE and RUNNING.

a) the detection of when a norm is active,
b) the detection of a violation on a norm, and
c) the handling of the violations.

In this section we are going to focus on the detection mechanisms, as they are central in the enforcement of norms. We talk more about violations, sanctions and repairs in §3.4. In order to support the task of agents enforcing norms, the agent platform should provide time-efficient services to help those agents to enforce proper behaviour in large agent societies.

- **Detection of the occurrence of an action.** In the case of agent actions, there are three possible points to be detected: a) when the action is going to be performed, b) it is being performed, or c) it is done. In an agent platform with several agents performing different actions at the same time, a question arises on how to implement the detection of the occurrence of actions. The agents enforcing norms may become overloaded on trying to check any action on any time. Therefore in [22] we proposed to create two plaftorm mechanisms: a) a **black list mechanism** of actions to be checked, and b) an **action alarm mechanism** that triggers an alarm when a given action A on the black list attempts to start, is running or is done. This trigger mechanism has to do no further checks, only to make the enforcer agent aware of the occurrence of the action. The action alarm mechanism can only be done with actions defined in the institutions' ontology, which specifies the way each action is to be monitored. For instance, when the performance of the action $assign(organ, recipient)$ should be checked, the action is registered by an enforcer agent on the black list. Then as soon as $assign(organ, recipient)$ occurs, the trigger mechanism sends an alarm to the enforcer, that will check if the action was legal or illegal given the norms for that context.
When actions are performed by users through an user interface, the action alarm mechanism can be placed in the interface itself. In the case of the following norm:

$$\mathsf{PERMITTED}(administrator\ \mathsf{DO}\ include(Suspect\_Data, Criminal\_Register))$$

The inclusion of the personal data of the suspect is done by all users through a special form. Therefore the interface knows when the user is filling in suspect data, and at the moment of submission of such data to the system can send an alarm to the enforcer.
- **Detection of activation and deactivation of norms**. In the case of conditional norms we have to detect the *activation of the norm* (when condition C is true) and *the deactivation of the norm* (when predicate P or action A is fulfilled or C does not hold). An additional issue is to establish the allowed *reaction time* between the activation and deactivation of an obligation, i.e. the time that is allowed for the completion of the obligation when it becomes active (e.g. immediately, in some minutes). [6] The length of the reaction time for each norm is highly dependent on the application domain. A violation does not occur when the norm becomes active but when the reaction time has passed. The manner and the moment to check the norm conditions is highly dependent on the verifiability levels of each check (see §3.3).
- **Deadlines.** Deadlines represent a special case in the implementation of conditional norms, as they are not that easy to check. Deadlines require a continuous check (second

[6] In theoretical approaches, the semantics are defined in a way that when an obligation becomes active, it has to be fulfilled instantly. But this is impractical for implementation, because agents need some time between detection and reaction.

by second) to detect if a deadline is due. If the institution has lots of deadlines to track, it will become computationally expensive. We propose to include within the agent platform a **clock trigger mechanism** that sends a signal when a deadline has passed. The idea is to implement the clock mechanism as efficiently as possible (some operating systems include a clock signal mechanism) to avoid the burden on the agents.

3.3 Verifiability levels

It is easy to see that a protocol or procedure satisfies a norm when no violations occur during the execution of the protocol. The real problem in norm checking lies, however, in determining when that violation occurs. For instance, in criminal investigations, a police officer should not have more (sensitive or private) information than needed for the investigation. So an officer is doing fine as long as no violation occurs (i.e. he does not have too much information). The real problem lies in determining when the officer actually has too much information.

Therefore, the implementation of the enforcement of norms is depending on two properties of the checks to be done: a) the checks being *verifiable* (i.e. a condition or an action that can be machine-verified from the institutional point of view, given the time and resources needed) and b) the checks being *computational* (i.e. a condition or action that can be checked on any moment in a fast, low cost way). Using these two properties, we can analyze their impact on the implementation of norm enforcement:

- **Norms computationally verifiable**: verification of all predicates and actions can be done easily, all the time. For instance:

  $$\begin{array}{c}\mathsf{PERMITTED}((user\ \mathsf{DO}\ appoint(regular\_user))\\ \mathsf{IF}\ (access\_level(user, register, `full\_control'))) \end{array}$$

  In this case it is clear that the verification can be easily done, because *authorization* mechanisms should be included on any multiagent platform to ensure security in open MAS.
  Implementation Guideline: In this case the verification can be performed each time that it is needed.
- **Norms not computationally verifiable directly, but by introducing extra resources**. In this case the condition or action is not directly (easily) verifiable, but can be so by adding some extra data structures and/or mechanisms to make it easy to verify. The *action alarm* and *clock trigger* mechanisms are examples of extra resources. For instance, in

  $$\begin{array}{c}\mathsf{OBLIGED}((buyer\ \mathsf{DO}\ bid(product, price))\\ \mathsf{BEFORE}\ (buyer\ \mathsf{DO}\ exit(auction\_house)))\end{array}$$

  checking that a buyer has done at least one bid in the auction house (i.e., checking all the logs of all the auction rounds) may be computationally expensive if there are no data structures properly indexed in order to check it in an efficient way (e.g. the agent platform keeping, for each buyer, a list of bids uttered, or having a boolean that says whether the buyer has uttered a bid). Another example is the following:

  $$\begin{array}{c}\mathsf{OBLIGED}((user\ \mathsf{DO}\ include(source(Suspect\_data), Criminal\_Register))\\ \mathsf{IF}\ (done(include(Suspect\_data, Criminal\_Register))))\end{array}$$

The detection of the inclusion of data is done by an *action alarm* mechanism placed in the user interface.
Implementation Guideline: include the extra data structures and/or mechanisms, and then do verification through them.

- **Non-computationally verifiable**: the check is too time/resource consuming to be done at any time.
  Implementation Guideline: verification is not done all the time, but is delayed, doing a sort of "garbage collection" that detects violations. There are three main families:
  - Verification done when the system is not busy and has enough resources.
  - Verification scheduled periodically. E.g. each night, once a week.
  - Random Verification (of actions/agents), like random security checkings of passengers in airports.
- **Observable from the institutional perspective, but not decidable**: That is, verifiable by other (human) agents that have the resources and/or the information needed. For instance:

  $$\mathsf{OBLIGED}((\mathit{register\_admin}\ \mathsf{DO}\ \mathit{correct}(\mathit{data}))\ \mathsf{IF}\ (\mathit{incorrect}(\mathit{data})))$$

  It is unfeasible for the system to check whether the information provided by users is incorrect without other sources of information. Therefore this check has to be delegated appropriately.
  Implementation Guideline: delegation of only those checks that cannot be performed by the system.
- **Indirectly observable from the institutional perspective**: These can be internal conditions, internal actions (like reasoning) or actions which are outside the ability of the system to be observed or detected (like sending a undetectable message between auctioneers in an auction).
  Implementation Guideline: try to find other conditions or actions that are observable and that may be used to (indirectly) detect a violation.

3.4 Violations, sanctions and repairs

Because there may be illegal actions and states which are outside the control of the enforcer, violations should be included in the normative framework. In order to manage violations, each violation should include a plan of action to be executed in the presence of the violation. Such a plan not only includes sanctions but also countermeasures to return the system to an acceptable state.

In order to link the norm conditions and the detection of activation and violation of norms with the violation management, we propose that a norm description includes, at least, the following:

- The *norm condition*: denotes when the norm becomes active and when it is achieved.
- The *violation state condition*: a formula derived from the norm to express when a violation occurs (e.g. for the norm $\mathsf{OBLIGED}((a, P)\ \ \mathsf{IF}\ C)$ this is exactly the state when C occurs and P does not, that is, the state where the norm is active, but not acted upon).
- The *violation detection mechanism*: a set of actions that can be used to detect the violation. This plan may include any of the proposed detection mechanisms described in §3.2.

- The *sanction*: the sanction is a plan (a set of actions) which should be executed when a violation occurs. This plan defines *punishment mechanisms*, either direct (fines, expulsion of the system) or indirect (social trust or reputation).
- The *repairs*: a contingency plan (set of actions) to recover the system from the violation.

An example (extracted from organ and tissue allocation regulations) is the following:

Norm condition	$\mathsf{FORBIDDEN}(allocator\ \mathsf{DO}\ assign(organ, recipient))$ $\mathsf{IF\ NOT}(allocator\ \mathsf{DONE}\ ensure\_compatibility(organ, recipient)))$
Violation condition	$\mathsf{NOT}(done(ensure\_compatibility(organ, recipient))\ \mathsf{AND}$ $done(assign(organ, recipient))$
Detection mechanism	$\{detect\_alarm(assign,' starting');$ $check(done(ensure\_compatibility(organ, recipient))); \}$
Sanction	$inform(board, "\mathsf{NOT}(done(ensure\_compatibility(organ, recipient))$ $\mathsf{AND}\ done(assign(organ, recipient))")$
Repairs	$\{stop\_assignation(organ);$ $record("\mathsf{NOT}(done(ensure\_compatibility(organ, recipient))\ \mathsf{AND}$ $done(assign(organ, recipient))", incident\_log);$ $detect\_alarm(ensure\_compatibility,' done');$ $check(done(ensure\_compatibility(organ, recipient)));$ $resume\_assignation(organ); \}$

4 Combining Norms

In previous sections we have focussed on the structure of one norm at a time. We have seen how the type and verifiability of a norm influences its implementation. However, when considering a set of norms (e.g. norms in a lawbook for a particular topic), there is an extra issue to handle, the defeasibility present in a set of norms, which has an important impact on implementation.

Defeasibility in a set of norms occurs when one or several norms can be defeated by the addition of more norms. There are two kinds of defeasibility, in this paper we will only focus on the latter:

- *Defeasibility of classification*: The semantic meaning of the concepts appearing in the norms may be extended, reduced or altered by the introduction of an extra set of norms.
- *Defeasibility of norms*: The impact and/or applicability of the obligations, permissions or prohibitions expressed in a given norm may be altered or even become unapplicable by the introduction of a extra set of norms introducing variations for some specific cases.

4.1 Defeasibility of norms

Norms in human regulations are formulated in a manner that is very similar to non-monotonic logics and default reasoning techniques [1] [15] [19]. That is, laws are generally specified in several levels of abstraction. On the most abstract level, normally the constitutional laws, a law defines the *default*, i.e. it defines what actions to take (or which predicates should hold) when certain conditions hold or specified situations arise. The “lower” levels of abstraction (e.g. applied law and decrees) generally specify exceptions to this default. They specify that certain situations do not follow the general norm and ask for a different approach.

Article 13

1. Any procurement that occurs directly through automated manner is recorded, as far as these procurements are not dispensed by decree of the Minister of Justice.
5. A procurement is not recorded in accordance with the first item, when it is a result of a linkage and a report of the linkage has been drawn up.

The example above is extracted from Dutch regulations on the management of Severe Criminality Registers. Article 13.1 specifies the obligation to record in the system log files any automated procurement of data that has not been stated in a decree from the Minister of Justice. This describes a quite clear situation, easy to be included in the decission making of the recording procedure of the system. We can express article 13.1 as follows:

$$\begin{array}{ll} A13.1 & \mathsf{OBLIGED}((\mathit{system}\ \mathsf{DO}\ \mathit{record}(\mathit{procurement}_i, \mathit{sys\_logs})) \\ & \quad \mathsf{IF}\ \mathsf{NOT}(\mathit{origin}(\mathit{procurement}_i, \mathit{decree}(Minister\_Of\_Justice)))) \end{array}$$

The addition of Article 13.5 suddenly *defeats* what is stated in Article 13.1, as it introduces a special, exceptional case where the first article does not hold. In principle we can express Article 13.5 as follows:

$$\begin{array}{ll} A13.5 & \mathsf{NOT}(\mathsf{OBLIGED}((\mathit{system}\ \mathsf{DO}\ \mathit{record}(\mathit{procurement}_i, \mathit{sys\_logs})) \\ & \quad \mathsf{IF}\ (\mathit{origin}(\mathit{procurement}_i, \mathit{linkage}_j)\ \mathsf{AND}\ \mathit{reported}(\mathit{linkage}_j, \mathit{sys\_logs})))) \end{array}$$

By this example we can see how defeasibility impacts in the reasoning process. There will be situations where both norms A13.1 and A13.5 will be triggered, and therefore two contradictory results (the obligation of recording and NOT the obligation of recording) appear. In this simple example it is quite clear that A13.5 overrides what is stated in A13.1 (by considering A13.1 the *default case* and A13.5 an *exceptional case*), but solving collisions at run-time for all possible combinations of norms is a complex and time-inefficient task.

Computational Guideline: Introducing the handling of defeasibility of norm sets in the reasoning mechanism is not a good option, as there is no efficient implementation of defeasible logics. Therefore there is a need to bypass defeasible reasoning, by solving all collisions off-line. Depending on the rate of changes in the law, there are two possible options to handle defeasibility of norms in implementation:

- *Changes in the law almost never occur*: As defeasible reasoning is computationally too complex, one possible option would be to avoid the defeasibility directly in the logical representation of the norms (i.e. the logical representation extracted from the human regulations re-structures the conditions for the base case and the exceptions in a way that it is not defeasible). In order to do so, the conditions that express when the exceptions occur should be introduced in the original norm as pre-conditions. For the previous example, expressions A13.1 and A13.5 can be fussioned in a single, non-defeasible expression as follows:

$$\begin{array}{ll} A13.1\_5 & \mathsf{OBLIGED}((\mathit{system}\ \mathsf{DO}\ \mathit{record}(\mathit{procurement}_i, \mathit{sys\_logs})) \\ & \quad \mathsf{IF}\ (\mathsf{NOT}(\mathit{origin}(\mathit{procurement}_i, \mathit{decree}(Minister\_Of\_Justice))) \\ & \quad \mathsf{AND}\ \mathsf{NOT}(\mathit{origin}(\mathit{procurement}_i, \mathit{linkage}_j) \\ & \quad \mathsf{AND}\ \mathit{reported}(\mathit{linkage}_j, \mathit{sys\_logs})))) \end{array}$$

The problems of this approach are that a) defeasibility should be completely handled by the designer or the knowledge engineer while building the computational representation, and b) there is no longer a direct mapping from each of the articles of the human law to the norm expressions in the computational representation, and therefore maintenance of the computational representation when there are changes in the law becomes highly difficult (e.g. what is to be changed in expression A13.1_5 if there is a new article that expresses an exception to the exception in Article 13.5?).

- *Changes in the law often occur (periodically)*: In this case the alternative is to build a defeasible computational representation of the norms, where each of the articles in the human law is mapped. In order to use the computational representation, an automated process seaches for those norms that become defeasible because of other norms and solves the problem by moving and/or adding conditions. The original defeasible representation of norms should include new objects in the object language to express the relations between expressions. For instance, Articles 13.1 and 13.5 could be represented as follows:

$$\begin{array}{ll} A13.1 & \mathsf{OBLIGED}((system\ \mathsf{DO}\ record(procurement_i, sys\_logs)) \\ & \mathsf{IF}\ (\mathsf{NOT}(origin(procurement_i, decree(Minister\_Of\_Justice))) \\ & \mathsf{AND}\ \mathsf{NOT}(\mathsf{CONDITIONAL\_EXCEPTION}(A13.1)))) \end{array}$$

$$\begin{array}{ll} A13.5 & \mathsf{CONDITIONAL\_EXCEPTION}(A13.1) \\ & \mathsf{IF}\ (origin(procurement_i, linkage_j) \\ & \mathsf{AND}\ reported(linkage_j, sys\_logs))) \end{array}$$

  In this case the representation explicitly specifies that expression A13.5 only impacts the conditions in expression A13.1. This information will be used by the automated process to generate the final, non-defeasible representation, getting automatically the expression A13.1_5 above.

  The advantage of this approach (that is a work in progress) is that each time there is a change in the law, the change can be easily made in the defeasible computational representation, which then automatically can be processed to eliminate defeasibility before its use.

5 Conclusions

In this paper we have focused on the operational aspects of institutional norms in MAS. We have looked at classifications of norms and have analyzed the verifiability of states and actions in norm expressions. We have also proposed and discussed several implementation guidelines on the enforcement of norms (i.e. detection and management) and several mechanisms that can be included to simplify norm enforcement on multiagent platforms.

We have presented a first draft of a machine-readable format for expressing complex norms, and using this format we have proposed a norm description, which includes the norm condition and violation detection and repair techniques, in order to make the first steps in implementing norm enforcement in MAS by means of violation handling.

We have discussed the problems of defeasibility of norms taht might arise when considering sets of norms.

Currently we are taking the first steps towards implementing the enforcement mechanisms presented here by introducing our norm model into ISLANDER. Besides that we are still examining how to mechanise the translation from defeasible norm sets to useable and implementable norms.

References

1. G. Antoniou. *Nonmonotonic Reasoning.* MIT Press, Cambridge, MA, 1997.
2. A. Artikis. *Executable Specification of Open Norm-Governed Computational Systems.* PhD thesis, Department of Electrical & Electronic Engineering, Imperial College London, November 2003.
3. G. Boella and L. van der Torre. Fulfilling or violating norms in normative multiagent systems. In *Proceedings of IAT 2004*. IEEE, 2004.
4. G. Boella and L. van der Torre. Normative multiagent systems. In *Proceedings of Trust in Agent Societies Workshop at AAMAS'04*, New York, 2004.
5. J. Broersen, F. Dignum, V. Dignum, and J.-J. Ch. Meyer. Designing a Deontic Logic of Deadlines. In *7th Int. Workshop on Deontic Logic in Computer Science (DEON'04)*, Portugal, May 2004.
6. C. Castelfranchi. Formalizing the informal?: Dynamic social order, bottom-up social control, and spontaneous normative relations. *Journal of Applied Logic*, 1(1-2):47–92, February 2003.
7. C. Castelfranchi, F. Dignum, C. Jonker, and J. Treur. Deliberative normative agents: Principles and architectures. In N. Jennings and Y. Lesperance, editors, *ATAL '99*, volume 1757 of *LNAI*, pages 364–378, Berlin Heidelberg, 2000. Springer Verlag.
8. F. Dignum, J. Broersen, V. Dignum, and J.-J. Ch. Meyer. Meeting the Deadline: Why, When and How. In *3rd Goddard Workshop on Formal Approaches to Agent-Based Systems (FAABS)*, Maryland, April 2004.
9. F. Dignum, D. Kinny, and L. Sonenberg. From Desires, Obligations and Norms to Goals. *Cognitive Science Quarterly*, 2(3-4):407–430, 2002.
10. V. Dignum and F. Dignum. Modeling agent societies: Coordination frameworks and institutions. In P. Brazdil and A. Jorge, editors, *Progress in Artificial Intelligence*, LNAI 2258, pages 191–204. Springer-Verlag, 2001.
11. V. Dignum, J.-J.Ch. Meyer, F. Dignum, and H. Weigand. Formal Specification of Interaction in Agent Societies. In *2nd Goddard Workshop on Formal Approaches to Agent-Based Systems (FAABS)*, Maryland, Oct. 2002.
12. M. Esteva, J. Padget, and C. Sierra. Formalizing a language for institutions and norms. In J.-J.CH. Meyer and M. Tambe, editors, *Intelligent Agents VIII*, volume 2333 of *LNAI*, pages 348–366. Springer Verlag, 2001.
13. M. Esteva, J.A. Rodríguez-Aguilar, B. Rosell, and J.L. Arcos. AMELI: An Agent-based Middleware for Electronic Institutions. In *Third International Joint Conference on Autonomous Agents and Multi-agent Systems*, New York, US, July 2004.
14. A Logical Framework for Ethical Behaviour between Infohabitants in the Information Trading Economy of the Universal Information Ecosystem (ALFEBIITE). `http://www.iis.ee.ic.ac.uk/~alfebiite/ab-home.htm`.
15. V. Lifschitz. Circumscription. In D. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 297–352. Clarendon Press, Oxford, 1994.
16. A. Lomuscio and D. Nute, editors. *Proc. of the 7th Int. Workshop on Deontic Logic in Computer Science (DEON04)*, volume 3065 of *LNCS*. Springer Verlag, 2004.
17. F. López y López and M. Luck. Towards a Model of the Dynamics of Normative Multi-Agent Systems. In G. Lindemann, D. Moldt, M. Paolucci, and B. Yu, editors, *Proc. of the Int. Workshop on Regulated Agent-Based Social Systems: Theories and Applications (RASTA '02)*, volume 318 of *Mitteilung*, pages 175–194. Universität Hamburg, 2002.
18. F. López y Lopez, M. Luck, and M. d'Inverno. A framework for norm-based inter-agent dependence. In *Proceedings of The Third Mexican International Conference on Computer Science*, pages 31–40. SMCC-INEGI, 2001.
19. J. McCarthy. Circumscription: A form of non-monotonic reasoning. *Artificial Intelligence*, 13(1-2):27–39, 1980.
20. P. Noriega. *Agent-Mediated Auctions: The Fishmarket Metaphor.* PhD thesis, Inst. d'Investigació en Intel.ligència Artificial, 1997.
21. J.A. Rodriguez. *On the Design and Construction of Agent-mediated Electronic Institutions.* PhD thesis, Inst. d'Investigació en Intel.ligència Artificial, 2001.
22. J. Vázquez-Salceda, H. Aldewereld, and F. Dignum. Implementing norms in multiagent systems. In G. Lindemann, J. Denzinger, I.J. Timm, and R. Unland, editors, *Multagent System Technologies*, LNAI 3187, pages 313–327. Springer-Verlag, 2004.