Implementing Norms in Multiagent Systems

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. There is a wide agreement on the use of *norms* in order to specify the expected behaviour of agents in open MAS. However, current norm formalisms focus on the *declarative* nature of norms. In order to be implemented, these norms should be translated into *operational* representations. In this paper we present our preliminary work on implementation of norm enforcement and issues on verifiability that highly affect this enforcement. We propose some mechanisms to be included in agent platforms in order to ease the 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 such an environment. Some foundational work in this direction has been done in the ALFEBIITE project [13], in particular in [1]. An Electronic Institution [9] [11] is a safe environment mediating in the interaction of agents. The expected behaviour of agents in such an environment is described by means of an explicit specification[1] of norms, which is a) expressive enough, b) readable by agents, and c) easy to maintain.

Current work on normative systems' formalization (mainly focused in Deontic-like formalisms [14]) 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 [8] [10] we have focused on the formal definition of norms by means of some variations of deontic logic that includes conditional and temporal aspects [4] [7], 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.

Implementing norms is not implementing a theorem prover that, using the norms semantics, checks whether a given interaction protocol complies with the

[1] The main reason for having explicit representations of norms is that norms may change over time. If norms are embedded in the agents' design and code, all the design steps have to be checked again and all the code verified to ensure compliance with new regulations.

norms. The implementation of norms should consider a) how the agents' behaviour is affected by norms, and b) how the institution should ensure the compliance with norms. The former is related to the *implementation of norms from the agent perspective*, by analyzing the impact of norms in the agents' reasoning cycle (work on this perspective can be found in [2] [3] [6]). The latter is related with the *implementation of norms from the institutional perspective*, by 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 [15] [16]. The framework aims to represent different kinds of agent societies based on norms. However, no implementation of the architecture applying it to a 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 analize 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, c) the different aspects of the norms to be specified, and d) who is responsible for norm enforcement.

We will 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 eCommerce, eGovernment or eCare domains) but also useful for implementation in MAS. Our implementation guidelines use the ISLANDER framework for institutions and platform as a starting point.

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 [5]. 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.

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, as well as implementation related issues. In §4 we discuss how violations are managed by means of plans of action. 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 [11] provides a formal framework for institutions [18] and has proven to be well-suited to model practical applications (e.g. electronic auction houses). 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 (or *illocutions* [17]) are structured through agent group meetings called *scenes* that follow well-defined protocols. This division of all the possible interaction among agents in scenes allows a modular design of the system, following the idea of other software modular design methodologies such as the Modular Programming or Object Oriented Programming. 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.

ISLANDER has been mainly used in *e*Commerce scenarios, and was used to model and implement an electronic Auction house (the *Fishmarket*). Furthermore, the AMELI platform [12] allows the execution of electronic institutions, based on the rules provided by ISLANDER specifications, wherein external agents may participate. The activity of these agents is, however, constrained by *governors* that regulate agent actions, to the precise enactment of the roles specified in the institution model.

ISLANDER provides a sound model for the domain ontology and has a formal semantics [18]. This is an advantage of its dialogical approach to organizations. However, in ISLANDER the normative aspects are reduced to the afore mentioned protocol 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 order to express complex norms we are going to use a language consisting of deontic concepts (OBLIGED, PERMITTED, FORBIDDEN) which can be conditional (IF) and can include temporal operators (BEFORE, AFTER). It is important to note that, although we do a formal analysis of norms in this section (which could be given a formal semantics as in some of our previous work [8] [10]), in this paper we are focusing on indicating possible implementation guidelines related with the different kinds of norms and the components in each of them.

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. Prevention of unwanted behaviour can only be achieved if there is full control over the addressee; otherwise, one should define and handle violations (see §4).

Before we look at how differences between the addressees of norms affect the implementation of enforcement mechanisms we will look at the types of norms that exists in human regulations.

3.1 Types of norms

In the legal domain, norms are descriptions of how a person (or agent) should behave in order to comply with legal standards. If we take a look at human regulations, we can observe three main types of norms:

– **Norms defining (refining) the meaning of abstract terms** (e.g. *"The criminal register administrator can be the Regional Police Force Commander, the Dutch National Police Force commander, the Royal Military Police Commander, the College of the Procurator-General or an official appointed by The Minister of Justice"*).
– **Norms defining (refining) an abstract action by means of subactions (a plan), a procedure or a protocol** (e.g. *"A request for examination [of personal data] [...] is sustainable after receipt of the payment of EUR 4,50 on account [...] of the force mentioning 'privacy request'"*)
– **Norms defining obligations/permissions/prohibitions**.

The first and second type of norms are only important in order to define the vocabulary to be used in a given regulation.[2] Work on law formalization focuses on the last kind of norms. We will also focus on the third type of norms in this paper.

[2] In an agent-mediated system, these norms would be implemented in the ontology of the system and/or in the refinement process of the actions on the system.

3.2 Addressee of norms

Although the amount of control over the addressee of a norm influences the operational semantics of the enforcement mechanisms, the detection of unwanted states or behaviour is necessary in both approaches. We distinguish 4 types of norms according to their addressee:

- **Norms concerning entities outside the scope and/or full control of the run-time system.** In this case no real enforcement can be done. This is usually the case of humans and/or outside agencies and systems which interact or affect the institution's behaviour but are outside of the full control. An example is an obligation to the user to provide correct data about a person. The system, having no other sources of information, has to trust the user.
  Implementation Guideline: receive some (external) information about the fulfillment of the norm. This information has to be trusted, because it cannot be directly checked.
- **Norms concerning external agents.** This is the group that has to be highly controlled and on who the majority of the enforcement has to be performed. However, we cannot see their internal mental states or reasoning process, or control it in any way. We can only see their observable behaviour in terms of (public) messages and (visible) actions.
  Implementation Guideline: enforcement depends on the verifiability of the predicates and actions that are present in the norms (see §3.4).
- **Norms concerning internal agents.** This is a group of agents that are internal to the system, performing facilitation tasks needed for the performance of the whole agent society.
  Implementation Guideline: enforcement of this kind of agents is similar to that of external agents, but as they are internal (i.e. built by the designers of the institution) we have some access and control to their internal states.
- **Norms concerning the major enforcers** (root enforcers). In this case enforcement should be done on the enforcers' behaviour. Enforcers are a special case of internal agents, so one possible option is to enforce their norms in a way similar to the internal agents. However, another question might then arise: How to enforce the enforcement on the enforcers? Since this *enforcement chain* can be extended *ad infinitum*, we should stop the enforcement chain somewhere. To achieve this we need to have full trust in our root enforcers, and therefore need full control over their internal architecture's design; their beliefs, goals and intentions.
  Implementation Guideline: 1) introduce the norms a) explicitly as goals and restriction rules interpretable by the enforcers or b) implicitly in the enforcer's code/design. 2) make sure that the environment/platform supports the enforcer on fulfilling the norms by a) providing enough information, and b) providing supporting enforcement mechanisms.

3.3 Norm expressions and enforcement

Not only the control over the addressee but also the elements present in the norm expressions (or norm condition) affect the enforcement of norms. Therefore, in this section, we first analyze norms depending on the elements that affect detection and we will discuss methods of enforcing these different kinds of norms. At the end of this section we will introduce a format for expressing the norm conditions. Note that we will use the term norms wherever Obligations ($\mathsf{OBLIGED}$) as well as Permissions ($\mathsf{PERMITTED}$) or Prohibitions ($\mathsf{FORBIDDEN}$) are meant.

For all the norms below, the implementation of enforcement is composed of three related processes 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 §4. It is also important to note that the precise moment to make the detection checks is highly dependent on the verifiability levels of each check (which we discuss in §3.4).

In the next sections we characterize norms 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.

Norms concerning that agent *a* sees to it that some condition/predicate *P* holds. In this case the norm is timeless, that is, the norm on the value of P is active at all times. There are three possible expressions:

$$\mathsf{OBLIGED}(a, P) \;\; \mathsf{PERMITTED}(a, P) \;\; \mathsf{FORBIDDEN}(a, P)$$

An example of such a timeless norm is the following:

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

Implementation Guideline: To determine whether the norm results in a violation we need to check whether P holds. Note that this check might be undecidable in general.

Norms concerning agent *a* performing an action *A*. In this case the norm on the execution of A is also timeless, that is, the norm is active at all times.

$$\mathsf{PERMITTED}(a \;\mathsf{DO}\; A) \;\; \mathsf{FORBIDDEN}(a \;\mathsf{DO}\; A)$$

There are no unconditional obligations ($\mathsf{OBLIGED}$), since this would express an obligation to execute an action all the time.[3] An example of an unconditional norm would be the following:

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

[3] In most cases, when such an obligation appears while modelling a human norm, it can be expressed better by means of a timeless obligation on a state. In other cases an implicit condition can be added to the norm for implementability reasons.

Note that action A can be an abstract action, that is, an action that is not present in the repertoire of the agents or defined in the protocol. In such cases A should be translated in more concrete actions to be checked.
Implementation Guideline: In the case of the unconditional $\mathsf{PERMITTED}$, we only have to check whether the agent has the correct role and whether all parametric constraints are met. In the case of the $\mathsf{FORBIDDEN}$ operator, we translate the abstract action A into concrete actions α and check on action α. In the case of computational verifiable actions, each one can be checked a) when the action is going to be performed, b) it is being performed, or c) it is done. We will call this process the *detection of occurrence of an action.*

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. Enforcer agents may become overloaded on trying to check any action on any time. We propose to create a) a *black list* of actions to be checked, and b) an *action alarm mechanism* that triggers an alarm when a given action A attempts to start, is running or is done. This trigger mechanism has to do no further checks, only to make the enforcer 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 $\mathit{bid}(\mathit{product}, \mathit{price})$ should be checked, the action is registered by an enforcer on the black list. Then as soon as $\mathit{bid}(\mathit{product}, \mathit{price})$ occurs, the trigger mechanism sends an alarm to the enforcer, that will check if the action was legal or illegal given the norms.

When actions are performed by users through a user interface, the action alarm mechanism can be placed in the interface itself. In the case of the following norm:

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

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 it can send an alarm to the enforcer.

Norms concerning a condition P or an action A under some circumstance C. The norm is conditional under C. This means that 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 time span 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). 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. This *reaction time* is ignored in norm theories, but has to be addressed when implementing norms. 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.[4]

A condition C may be a) a predicate about the state of the system, or b) a state of some action (starting, running, done).

$$\begin{array}{ll} \mathsf{OBLIGED}((a,P)\ \mathsf{IF}\ C) & \mathsf{OBLIGED}((a\ \mathsf{DO}\ A)\ \mathsf{IF}\ C) \\ \mathsf{PERMITTED}((a,P)\ \mathsf{IF}\ C) & \mathsf{PERMITTED}((a\ \mathsf{DO}\ A)\ \mathsf{IF}\ C) \\ \mathsf{FORBIDDEN}((a,P)\ \mathsf{IF}\ C) & \mathsf{FORBIDDEN}((a\ \mathsf{DO}\ A)\ \mathsf{IF}\ C) \end{array}$$

An example is the following:

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

Implementation Guideline: In the case of OBLIGED, the implementation of the enforcement depends on the verifiability of the condition C (detection of the activation of the obligation) and then the verifiability of P or A (detection of the deactivation of the obligation) In the case of enforcement of a Permission (PERMITTED) or a Prohibition (FORBIDDEN) such as $\mathsf{PERMITTED}((a\ \mathsf{DO}\ A)\ \mathsf{IF}\ C)$, the order of the checkings should be reversed: first detect the occurrence of the action A or the predicate P, and then check if condition C holds. Detection of occurence of an action A is done again with the *black list* and *action alarm* mecanisms.

Conditional norms with deadlines. This is a special type of conditional norm where the start of the norm is not defined by a condition but by a deadline. We distinguish two types of deadlines:

- Absolute deadline (hh:mm:ss dd/mm/yyyy). E.g. 23:59:00 09/05/2004.
- Relative deadline: a deadline relative to an event C' (time(C') +/- lapse) E.g. $time(done(bid)) + 5min$

There are 12 possible expressions with deadlines, by combining the three deontic operators, the temporal operators (BEFORE and AFTER) and applying them to actions or predicates. Examples of such expressions are:

$$\begin{array}{ll} \mathsf{OBLIGED}((a,P)\ \mathsf{BEFORE}\ D) & \mathsf{PERMITTED}((a\ \mathsf{DO}\ A)\ \mathsf{AFTER}\ D) \\ \mathsf{FORBIDDEN}((a,P)\ \mathsf{BEFORE}\ D) & \end{array}$$

Implementation Guideline: In the case of permissions (PERMITTED) and prohibitions (FORBIDDEN), the procedure is as in conditional norms: first detect the occurence of the action or the predicate, and then check the deadline. It is important to note that there is a relationship between permissions and prohibitions:

$$\begin{array}{l} \mathsf{PERMITTED}((a,P)\ \mathsf{BEFORE}\ D) \Leftrightarrow \mathsf{FORBIDDEN}((a,P)\ \mathsf{AFTER}\ D) \\ \mathsf{FORBIDDEN}((a,P)\ \mathsf{BEFORE}\ D) \Leftrightarrow \mathsf{PERMITTED}((a,P)\ \mathsf{AFTER}\ D) \end{array}$$

[4] Note that this also holds for unconditional norms.

In the case of OBLIGED, the deadline should be checked first, and then the occurence of A or P is verified. But deadlines are not that easy to check. They require a continuous check (second by second) to detect if a deadline is due. If the institution has lots of deadines 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.

Obligations of enforcement of norms. In this case the norms concerning agent b generate obligations on agent a.

$$\begin{array}{l}\mathsf{OBLIGED}(a\ \mathsf{ENFORCE}(\mathsf{OBLIGED}(b...)))\\ \mathsf{OBLIGED}(a\ \mathsf{ENFORCE}(\mathsf{PERMITTED}(b...)))\\ \mathsf{OBLIGED}(a\ \mathsf{ENFORCE}(\mathsf{FORBIDDEN}(b...)))\end{array}$$

Implementation Guideline: When a is an internal enforcer, as we have full control on internal agents, we implement this norm by placing the enforcement as a goal of the agent (as we discussed in §3.2). When a is not an internal enforcer but an external agent and the system has to enforce that a enforces another agent b's norms, we have two enforcement mechanisms:

– a enforces the norm on b: in this case, depending on b's norm, a has to detect the start and the end of the norm, and the occurrence of a violation, as explained in previous sections. In the case of a violation, a should execute the plan of action defined to solve such a situation.
– *root enforcer* enforces the obligation of a: in this case a root enforcer should detect those situations when b has violated its norm and a has not executed the plan of action to counteract the violation. The safest way would be to have a root enforcer closely checking the behaviour of b just as a should do, detect the start and the end of b's norm and the occurrence of violations, and then verify that a properly executes the plan of action. However, this is computationally expensive (we have two agents doing the same enforcement). If we want to have a safe enforcement we should use an internal agent to do it. Otherwise, if we have delegated some enforcement to agent a, we should not spend lots of resources on verifying a's behaviour. In this case the best option is, depending on the verifiability of the checks, to do some of the checks randomly or when the system has enough resources to detect violations that have not been counteracted.

Norm Condition Expression Language. Using the different kinds of norms that we discussed in the previous sections we can now specify a generic language for expressing norm conditions. This language was already used to express the examples in the previous sections. Although this language can be given a formal semantics, we refrain from doing so for now, but refer to [7] [10].

Definition 1 (Norm Condition).

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

Definition 1 shows that norm conditions can either be concerning states, e.g. for a norm such as "*buyers should not have a negative saldo*", or concerning actions, e.g. for norms like "*administrators are allowed to include personal data concerning suspects in the Criminal Register*". The definition allows the norm condition to be conditional, allowing the expression of norms like "*one should include the source of the data when including suspect data in the Criminal Register*", as well as norm conditions including temporal aspects in the form of deadlines, for instance "*personal information in a Criminal Register is to be deleted when no new data has been entered within the last five years proving that the data is necessary for the investigation*". 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.

3.4 Verifiability levels

Now that we know what kinds of norms there are, we have to investigate how to use this information in order to enforce norms. 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). We distinguish between the following three levels of verifiability:

[5] The conditions (C) and propositions (P) are expressed in some kind of propositional logic. This logic can use deontic (cf. [8] [10]), or temporal (cf. [4] [7]) operators. Note however that this logic should at least include some operational operators like, for instance, DONE and RUNNING.

- **Computationally verifiable**: a condition or action that can be verified at any given moment.
- **Non-computationally verifiable**: a condition or action that can be machine-verified but is computationally hard to verify.
- **Non-verifiable**: a condition or an action that cannot be verified from the system (the institution) point of view, because it is not observable.

Using these levels we can look at 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:

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

  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

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

  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:

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

  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:

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

  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.
- **Not verifiable at all**: Should not be checked, because, e.g. it is completely unfeasible to do so (placed here for completeness, but no example found).

4 Violations, sanctions and repairs

As described in §3, we cannot assume to have full control over the addressees. 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 (repairs).

In section 3.3 we have introduced a machine-readable format for expressing norm conditions, and have discussed how to detect the activation and violation of norms. In order to link these detections with the violation management, we propose that a norm description includes, at least, the following:

- The norm condition (expressed as seen in §3.3).
- The violation state condition.
- A link to the violation detection mechanism.
- A sanction: the sanction is a plan (a set of actions) to punish the violator.
- Repairs: a plan (set of actions) to recover the system from the violation.

In this format, the *norm condition*-field is denoting when the norm becomes active and when it is achieved. The *violation* is 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 *detection mechanism* is a set of actions that can be used to detect the violation (this includes any of the proposed detection mechanisms described in §3.3). The set of actions contained in the *sanction*-field is actually a plan which should be executed when a violation occurs (which can contain imposing fines, expulsing agents from the system, etc.). Finally, the *repairs* contains a plan of action that should be followed in order to 'undo' the violation. Definition 2 show how these elements make up the norm.

Definition 2 (Norms).

$$\begin{aligned}
\mathsf{NORM} &:= \mathsf{NORM\_CONDITION}\\
&\ \mathsf{VIOLATION\_CONDITION}\\
&\ \mathsf{DETECTION\_MECHANISM}\\
&\ \mathsf{SANCTION}\\
&\ \mathsf{REPAIRS}\\
\mathsf{VIOLATION\_CONDITION} &:= \mathit{proposition}\\
\mathsf{DETECTION\_MECHANISM} &:= \{\mathit{action\ expressions}\}\\
\mathsf{SANCTION} &:= \mathsf{PLAN}\\
\mathsf{REPAIRS} &:= \mathsf{PLAN}\\
\mathsf{PLAN} &:= \mathit{action\ expression} \mid \mathit{action\ expression}\ ;\ \mathsf{PLAN}
\end{aligned}$$

For the formal definition of NORM_CONDITION see definition 1 in section 3.3.

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

Norm condition	$\mathsf{FORBIDDEN}(allocator\ \mathsf{DO}\ assign(organ, recipient))$ $\mathsf{IF}\ \mathsf{NOT}(hospital\ \mathsf{DONE}\ ensure\_compatibility(organ, recipient)))$
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	$\{stop\_assignation(organ);$ $record("\mathsf{NOT}(done(ensure\_compatibility(organ, recipient))\ \mathsf{AND}$ $done(assign(organ, recipient))", incident\_log);$ $detect\_alarm(ensure\_compatibility, 'done');$ $check(done(ensure\_compatibility(organ, recipient)));$ $resume\_assignation(organ);\}$

This example shows how violations and their related plans of action are defined. The violation condition defines when the violation occurs in terms of concrete predicates and actions (in the example, the violation condition uses exactly the predicates and actions in the norm expression as there is no need to refine them). The detection mechanism is defined as a plan (in this case involving an *action alarm* mechanism detecting each time that an assignment is attempted). Sanction plans define *punishment mechanisms*, either direct (fines, expulsion of the system) or indirect (social trust or reputation). In this scenario the punishment mechanism is indirect, by informing the board members of the transplant organization about the incident. Finally, the repairs is a plan to solve the situation (that is, a contingency plan). In action precedence norms (A precedes B), it usually has the same structure: stop action B (*assign*), record the incident in the systems' incident log and then wait (by means of the action alarm mechanism) for action A (*ensure_compatibility*) to be performed.

5 Conclusions

In this paper we have focused on the operational aspects of institutional norms in MAS. We have analized the problem by categorizing norms depending on actors involved, verifiability of states and actions in norm expressions, and temporal aspects. Then we have proposed some implementation guidelines on the enforcement of norms (i.e. detection and management) and the inclusion of some mechanisms (*black lists*, *action-alarms*, *clock-triggers*, *authorization*) to simplify norm enforcement on multiagent platforms.

We have also presented a first draft of a machine-readable format for expressing complex norms, like the ones appearing in domains such as *e*Commerce, *e*Government and *e*Care. Using this format we have proposed a norm description, which includes the norm condition and violation detection and repair techniques, in order make the first steps in implementing norm enforcement in MAS by means of violation handling.

Currently we are taking the first steps towards implementing the enforcement mechanisms presented here by introducing our norm model into ISLANDER, and adding the proposed enforcement mechanism to the E-INSTITUTOR platform.

References

1. A. Artikis. *Executable Specification of Open Norm-Governed Computational Systems*. PhD thesis, Department of Electrical & Electronic Engineering, Imperial College London, November 2003.
2. G. Boella and L. van der Torre. Fulfilling or violating norms in normative multiagent systems. In *Proceedings of IAT 2004*. IEEE, 2004.
3. G. Boella and L. van der Torre. Normative multiagent systems. In *Proceedings of Trust in Agent Societies Workshop at AAMAS'04*, New York, 2004.
4. 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.
5. 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.

6. 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.
7. 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.
8. F. Dignum, D. Kinny, and L. Sonenberg. From Desires, Obligations and Norms to Goals. *Cognitive Science Quarterly*, 2(3-4):407–430, 2002.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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`.
14. 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.
15. 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.
16. 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.
17. P. Noriega. *Agent-Mediated Auctions: The Fishmarket Metaphor.* PhD thesis, Inst. d'Investigació en Intel.ligència Artificial, 1997.
18. J.A. Rodriguez. *On the Design and Construction of Agent-mediated Electronic Institutions.* PhD thesis, Inst. d'Investigació en Intel.ligència Artificial, 2001.