Dynamic assignment of roles, rights and responsibilities in normative multi-agent systems

FARNAZ DERAKHSHAN, *Department of Electronic and Computer Engineering, University of Tabriz, 29th Bahman St, Tabriz, Iran.*
E-mail: derakhshan@tabrizu.ac.ir

TREVOR BENCH-CAPON and PETER MCBURNEY, *Department of Computer Science, University of Liverpool, Liverpool L69 7ZF, UK.*
E-mail: tbc@liverpool.ac.uk; mcburney@liverpool.ac.uk

Abstract

Open multiagent systems (MAS) typically require the participating agents to comply with system-level rules, or *norms*, and may punish non-compliance. An interesting challenge for designers of such systems is how to provide for dynamic, run-time allocation of norms. In this article, we present a novel, framework-independent approach enabling a norm architecture to be overlaid on any type of multiagent system. We achieve this by defining rights, responsibilities and sanctions in a manner which allows for their run-time assignment and re-assignment to agents in a normative multiagent system. The overlay architecture we propose requires the system to include a normative knowledge (KB) base containing explicit representation of norms as conditional rules, where the preconditions for instantiation of a norm are various run-time occurrences, such as the execution of some agent action, or the occurrence of some event. Our method has both wide generality and flexibility, allowing at one extreme few or no agent roles, each role having many conditional norms, or allowing, at the other extreme, many roles, each having only a small number of conditional norms, or allowing, indeed, any position between these two extremes.

1 Introduction

The rise of distributed computer systems and the growth of the Internet has led to a re-conceptualization of computer systems as comprising interacting software entities, called *agents*, each with a greater or lesser degree of decision-making autonomy [22]. Such multiagent systems (MAS) have been developed and implemented for domains as diverse as finance, insurance, manufacturing, telecommunications, medicine and the management of civil emergencies [23]. Initially, most deployed MAS comprised software agents representing people or organizations employed by a single organization, or a closed-user group of closely-connected organizations. The MAS for cross-national motor vehicle insurance claims reconciliation described in [31], for example, comprises agents representing several national insurance companies engaged as partners in business together. Increasingly, however, MAS are being created as open systems—in other words, systems which allow any agent, representing any person or organization, able to interface to the system to join and participate. Such systems require considerably more design flexibility, since the intentions and behaviours of unknown participants is harder to predict than for known participants. In addition,

Vol. 23 No. 2, © The Author, 2011. Published by Oxford University Press. All rights reserved.
For Permissions, please email: journals.permissions@oup.com
Published online August 31, 2011 doi:10.1093/logcom/exr027

systems comprising agents designed or implemented by different design teams need to allow for non-standard decision procedures, malicious or whimsical objectives, unruly or unexpected behaviours, and bug-ridden code.

These considerations have led researchers and software engineers in MAS to consider systems in which the participants are subject to rules of conduct, or what are loosely called norms. One important distinction to make in such systems is between *norm regimentation* and *norm enforcement*. Norm regimentation ensures that any violation of the system rules is impossible, with participating agents given no autonomy to violate a norm. Under a regime of norm enforcement, in contrast, some or all participating agents are able to decide whether to conform or not to the system norms, with potential penalties if they choose not to conform. Given that system designers usually have a great deal of freedom in designing systems, one could ask why designers would ever allow agents to violate the norms of the system. A key reason is that desired norms may conflict with one another, and a choice may therefore need to be made between conflicting norms. The most appropriate norm to be applied may vary from one situation to another, and may even depend on the attributes or identity of the agents involved in the situation. A system designer may not be able or may not wish to specify the preferred choices in all these situations, and may prefer to leave the decision to be made by the participating agents on a case-by-case basis.

For example, perhaps the most sophisticated approach to the design and engineering of rule-governed MAS to date is the paradigm of Electronic Institutions (EI) of the Spanish national Artificial Intelligence Research Institute (IIIA) in Barcelona [3]. This approach provides an over-arching framework to regulate the conduct of software agents in a multiagent system, particularly for open systems. Electronic institutions aim to explicitly define the rules of permissible or prohibited actions in an agent society in a manner similar to that of human social institutions. Implementation and deployment of systems based on this model have been facilitated by the creation of software design and engineering tools to develop electronic institutions [9]. Although the current implementation of Electronic Institution design engineering tools provide an interface for defining obligation norms for agents external to the system, in practice the actions of such external agents are filtered by agents internal to the system, called *governors*. Vázquez-Salceda *et al*. have explained the weaknesses of this approach of norm regimentation in Electronic Institutions in detail in [32]. In consequence, several researchers have developed extensions to the Electronic Institution framework to create a language for expressing norms, as in [12, 13, 32], or mechanisms for norm violation detection and responses, as in [1].

This work extending the Electronic Institutions framework is leading towards norm enforcement regimes, possibly with dynamic, run-time allocation of norms, although we do not believe it is quite there yet. In contrast, we have adopted a more generic, framework-independent approach to enable a norm architecture to be overlaid on any type of multiagent system. We achieve this by defining rights, responsibilities and sanctions in a manner which allows their run-time assignment and re-assignment to agents in a normative multiagent system. The overlay structure requires the system to include a normative knowledge base (KB) containing rule-based norms, and these norms may be main norms (i.e., rules of behaviour for agents participating in the system), enforcement norms, norms for violation detection, and norms for violation responses, such as punishments and rewards. We present the details of this structure in this article. In the next section, we present a discussion of norms, roles, rights and responsibilities, as necessary prelude to the main work of the article. This discussion is not intended to be a contribution to a theory of norms, but simply background material for understanding of the concepts used later in the article. Section 3 then presents a method we propose for normative MAS which will enable the dynamic allocation of roles, rights and responsibilities, and Section 4 presents a system architecture for implementation of this method which may be overlaid on an

existing MAS. The article ends with a discussion of related and future work in Section 5. We are assuming throughout this article that the MAS under consideration may be open—that is, the systems may permit externally-designed agents to enter and leave, more or less at will. A key consequence of this assumption is that, by definition, any norms enforced within the system are accepted voluntarily by the agents participating in the system, since these agents can always '*vote with their feet*', deciding to not enter, or to leave, the system. Since all norms are assumed to be accepted voluntarily by the participating agents, we have therefore not modelled norm acceptance, a subject of recent discussion in the agent literature, e.g. [14, 21].

2 Norms, roles, rights and responsibilities

2.1 Norms in general

Autonomous agents are generally presumed to act in accordance of what they perceive to be their own interests. Their actions, however, affect both the state of the world and other agents, both directly and because the outcome of the action of an agent depends on what other agents do in the situation. What is in the best interests of one agent may be detrimental to other agents, and to the common good, whether conceived of as an aggregation of individual goods, or as supervening on individual goods. For these reasons it is desirable to constrain, or at least influence, the choices of autonomous agents in a society.

For this reason actions are generally subject to norms. Norms state what is obligatory (O), forbidden (F) and permitted (P) for agents in that society. Norms have many sources: for example, we have universal norms, moral norms, informal societal norms, explicit legal norms and explicit norms associated with particular organizations and groups. Here we will be concerned only with explicit norms. These norms are promulgated by some authority, be it a Government, an employer, a club or an institution such as a library. Agents subject themselves to such norms by being citizens of a particular state, by accepting particular employment, or enrolling in a society or institution. In all cases except the first they agree to abide by the norms, relinquishing some autonomy for the sake of the benefits of participating in the institution. We will be particularly concerned with these voluntarily accepted norms associated with institutions.

The various normative modalities are interdefinable. Thus, saying that it is obligatory for some agent to undertake action α is equivalent to saying that it is forbidden for that agent not to undertake α, and that this in turn is equivalent to saying that it is not permitted for that agent not to undertake α. In symbols:

$$O\alpha \Leftrightarrow F\neg\alpha \Leftrightarrow \neg P\neg\alpha$$

But what do norms qualify? A distinction is often drawn between *ought to be*, in which it is obligatory to bring about a particular state of affairs, and *ought to do*, where some particular action is obligatory. Both certainly occur, but it is more usual for norms to apply to actions rather than states of affairs. The Ten Commandments, to take a classic example, comprise nine prohibitions and an obligation, all relating to actions. On the other hand, while certain actions may be undesirable in themselves, norms are typically justified in the consequences the actions would have. Thus the norm *it is forbidden to walk on the grass* is designed to ensure that the grass remains in good condition, rather than to prevent the action for its own sake. Why then do norms usually relate to actions? One reason is that is hard to envisage the consequences of actions: it is more effective to proscribe walking on the grass than demanding that agents ensure that it remains in good condition. Moreover actions are often uncertain in their effect and so it is simpler for an agent to refrain from an action than to

ensure that it does not act so as enter a particular state. Finally applying norms to states of affairs is insufficiently limited: given a norm that *the grass must remain in good condition*, agents might feel obliged not only not to walk on it themselves, but to prevent others from walling on it, and perhaps even to water and fertilize it, destroy weeds and pests and so forth. One important feature of norms is that agents should be aware of what is required of them, so that they can comply with the norms, and be sure that they have complied with them.

Although the deontic modalities have some similarities with the alethic modalities, there are very important differences, even when the α of $O\,\alpha$ is a proposition. In particular, it is not the case that $O\,\alpha \implies \alpha$. Norms can be violated: agents choose whether or not to comply with them. And typically there will be circumstances in which a norm is more honoured in the breach than the observance. Sometimes, in complex normative systems, norms will conflict and so it will not be possible to comply with all of them. But even where there is no conflict, it may still be right to violate a norm: it is obligatory to drive motor vehicles on the left side of the road in the UK, but if the way is blocked by a landslip, it would be absurd to hit the obstacle or to wait for its removal rather than driving on the right, with due care and attention, to get past it. Even a universal and deeply entrenched norm such as that against killing permits of exceptional circumstances in which its violation will be seen as desirable. An autonomous rational agent must be allowed to violate norms: obsessive adherence to them is the mark of unthinking petty officials and certain psychological disorders.

2.2 *Types of norms*

We begin by distinguishing between several different types of norms, including coordination norms, constitutive norms, procedural norms, cooperation norms, and substantive norms. We present this broad classification in order to guide the subsequent discussion in the article; the classification is not intended to be mutually-exclusive or exhaustive. The literature on norms in philosophy, legal theory and logic is extensive, (for example, see: [5, 15, 17, 20, 28, 29]) and the classification here is not intended as a contribution to that theory.

Coordination norms serve to ensure that the actions of agents are aligned, to improve the information available to agents and so to inform their choices. A good example is the norm that *cars should be driven on the left hand side of the road*. What is important here is that all agents do the same thing, and are enabled to do this by having good expectations about what other agents will do. It doesn't matter which side of the road is chosen by the norm—countries other than the UK make the opposite choice with equally good results. Such norms may be violated in certain cases (an example was given earlier), provided that care is taken that the violation will not confound the expectations of other agents. In normal circumstances violation will never be in the interests of an agent. Interestingly in this case if any agent does violate the norm it is better if all agents violate it than if only a few do. Conventions, as in the work of Lewis [20], may be seen as examples of co-ordination norms.

Constitutive norms are usually associated with institutions and normally relate to acts meaningful within that institution [28]. They state what will *count as* an institutional act. Thus in an auction certain gestures may count as bidding. Such norms are essential for the operation of the institution and cannot be violated. Either an agent complies with the norms for making a bid, or no bid is made. It is therefore always in the interests of agents to follow the norm if they wish to perform the institutional act. Such norms are like the laws of Chess: they must be followed or one ceases to play Chess.

Procedural norms are similar, in that they state a procedure for performing an act. For example Social Security Law states what procedure should be followed in making a claim. These are similar to coordination norms, in that they inform the agent what is needed to achieve a goal, and similar to constitutive norms, in that compliance with them assures the agent that the desired act has been performed. They can, however, be violated: it may be that a claim not following the correct procedure is never the less accepted, or it may in some cases be desirable to cut corners and omit parts of the procedure.

Cooperation norms govern situations in which it may not be in the interests of an individual to act in a certain way, but where it is beneficial if all agents do act in this way. Taxation provides an example. Any individual agent will be better off not paying taxes, but as a group agents will be worse off if none pay taxes. Here violation is tempting, especially since the benefits typically accrue to all agents provided enough comply.

Finally substantive norms govern actions which while they may be in the interests of the agent concerned will be damaging to other agents or the community in general. Laws against theft provide an example. Here temptation is also great: the agent complying with the norm obtains no benefit, but should comply for the good of others.

Thus in several cases there is a temptation to violate the norm, even when there are no exceptional circumstances to justify non-compliance in any terms other than self interest. For this reason norms are typically backed by sanctions. We may distinguish two types of sanctions: those designed to deter agents from violations, and those designed to ensure that no harm results from violation. The first sort are familiar as punishments, and should be of a sufficient severity that they cancel out any gains that may accrue to the agent. For example a heavy fine for non-compliance with taxation laws should (provided the violations are normally detected and the sanctions regularly applied) be sufficient to make compliance in the self interest of the agents. The second type may be illustrated by library fines. Although it is proper to return books on time, failure to do so can be offset by an addition to the library's revenue. Indeed, if the book is not in demand, it may even be preferable if it is kept too long by a borrower so that it makes money for the library rather than gathering dust on the shelf. Many sanctions involving compensation similarly are designed to remove or at least ameliorate the harm done by the violation.

2.3 *Roles*

Some norms apply to everyone:

$$F\alpha \Longrightarrow \forall x F x_{\alpha}$$

Thus *it is forbidden to walk on the grass* means that Tom, Dick and Harry are all forbidden to walk on the grass. In other cases, particularly in organizations and institutions, the norms are applicable to individuals only when they are performing particular roles. In a club officers may be subject to norms not applicable to ordinary members. For example, the secretary may be obliged to distribute an agenda a week before the annual meeting. The norm will apply to whoever fulfils that role: it is not that Harry is obliged to distribute the agenda, but that whoever is the secretary is so obliged, which may at a particular time be Harry. In such cases taking on a role brings with a number of additional norms.

Role-related norms are often bound up with procedures. In formulating procedural norms, often reference is made to roles relating to that procedure. Thus when claiming a benefit the *claimant* has certain duties, such as providing true information, and the *adjudicator* has certain different duties, such as resolving the claim within a certain time.

2.4 Rights and responsibilities

Generally, being the subject of a norm, whether a universal norm or one applicable in virtue of fulfilling a role, can be seen as imposing certain responsibilities on an agent. If an agent is obliged, respectively forbidden, to do ϕ, then that agent has the responsibility to, respectively not to do, ϕ. Very often associated with this responsibility will be a concomitant *right*. If agents are forbidden to steal, agents have a right not to be stolen from. These rights and responsibilities are also often related to roles: in the claiming benefit example, the adjudicator has the responsibility to decide the claim in a certain time, and the claimant has the right to have his claim decided within this time period. Specific obligations are often directed in this way: thus x has a responsibility to y that x ϕ, and y has the right that x ϕ. Under some circumstances, responsibilities and obligations may be able to be transferred or partially transferred to others, for example, via delegation. Much recent work in multi-agent systems has considered directed obligations, e.g. [16, 29], and delegation, e.g. [24, 25]. As with the classification of types of norms above, this article is not aiming to contribute to this literature, but simply to use these ideas for the framework presented in the next two Sections.

2.5 Norms in MAS

In an open MAS, norms provide a convenient way of attempting to produce desired behaviour. Typically in such a system the various agents involved will be fulfilling particular roles, and will be capable of actions which will produce both desired and undesired behaviour. If we model the system as a state transition diagram in a suitable formalizm, such as ATL [2], we can identify states which we wish to avoid, and paths we wish to follow, and forbid actions that will lead to undesirable states and oblige actions that will takes us to states we require. Imposing prohibitions is relatively easy, since we can simply disable the corresponding action for the appropriate agents. Imposing obligations is harder: even if we allow the obligated agent only one action, in an open system that agent may never the less not perform it. Moreover ensuring compliance in this way is not in the true spirit of norms, which accept that in certain circumstances violation is in fact desirable. An alternative approach is to make the agents aware of their rights and responsibilities and rely on their voluntary entry into the institution to motivate compliance, so that they can receive the benefits of membership, which presumably outweigh the costs of compliance. We may, however, still wish to guard against non-compliance by detecting violations and imposing sanctions, either to take the harm out of the situation, or to deter violation by making the costs to the agent outweigh the benefits once the sanction is taken into account.

3 Dynamic assignment

3.1 Requirements for dynamic assignment

Two main elements must be specified in order to assign rights and responsibilities to agents in a normative MAS dynamically. First, the features of norms (and their related rights and responsibilities and sanctions) associated with the roles of the normative MAS need to be represented and stored in a normative KB. Secondly, the sources of change in the MAS which determine which norms are applicable and hence which rights, responsibilities and sanctions apply at a given time need to be specified. Therefore, dynamic assignments of norms to agents in a normative MAS are founded on two main elements: a source of norms potentially applicable to roles and a source of dynamism. These aspects are more fully described in [8].

Normative KB: the normative KB stores all the norms associated with roles in the normative multi-agent system. Using a descriptive normative language all norms associated with roles can be formally defined. One of us has proposed [7] a normative language in which we can describe the different types of norms found in legal systems including: various types of legal modalities, such as obligation, prohibition, permission and right [19]; enforcement modalities, such as punishment, reward and compensation [30, 32]; and several key elements of norms such as addressee [19], beneficiary [16], temporal notions such as currency and deadlines [19, 32] and preconditions activating norms [32].
Sources of dynamism: the sources of dynamism and change in a system are assumed to comprise:

- variations in the agent population as agents enter or leave the system;
- the occurrence of actions performed by agents and other system events;
- changes in environmental parameters, such as price in an auction system; and
- important threshold times being reached.

When a source of dynamism causes a change in the MAS we say that a *run-time occurrence* takes place.

As the result of a run-time occurrence, the conditions activating one or more norms of the normative KB may become satisfied, and so rights, responsibilities and sanctions need to be assigned to one or more agents. Therefore, we can use the run-time occurrences as the triggers to instantiate a dynamic assignment or re-assignment of normative consequences to agents in a system. For example, suppose we have the following two norms in an auction system:

Norm1: The Auctioneer is obliged to reject all but the highest bids during the auction session.

Norm2: During an auction session, if a lower bid is placed and not rejected by the Auctioneer, punishment2 will be applied to the Auctioneer.

According to Norm1, the obligation is activated and assigned to the Auctioneer agent only when other agents perform certain specified actions during the auction session. Norm2 states that if the auctioneer violates Norm1, then the Auctioneer will be punished. So if the precondition of Norm1 is satisfied, Norm1 will be activated and the resulting obligation placed on the Auctioneer; the Auctioneer can discharge this obligation by rejecting the bid. As a result, the activation of Norm2 is subject to temporal conditions (only during the auction session) and event conditions (placement of a lower bid by another agent), while deactivation of Norm2 is subject to action conditions (rejection of the lower bid by the Auctioneer). In this way, the activation and deactivation of each specific norm can occur dynamically at run-time, and so assigning the associated rights and responsibilities of the activated norms to the relevant agent will also be a dynamic task.

3.2 Roles versus rights and responsibilities

We are chiefly interested in norms which are attached to roles, and the relationships between them. If we had no roles, all rights and responsibilities would have to be expressed using conditional norms. The conditions in these norms would express two things: some would be intended to identify the role of the agent, and some would be intended to identify a particular situation in which the agent playing that role finds itself. For example, in a multiagent auction institution we might have a rule such as:

An agent which (1) is a member of the institution, (2) has made a bid in a particular auction, (3) has had the bid accepted by the auctioneer, (4) has made the highest bid in that auction, (5) when the auction closes shall pay the sum bid to the auctioneer.

Now this rule as a whole could define a situation in which any agent could find itself, a conditional norm that makes no reference to roles. Alternatively we could have a role *Institution Member*, defined

by condition (1), and conditions (2–5) would define the situation in which that agent must pay. Or we could define a role *Candidate Bidder* by conditions (1–2) and (3–5) would define the situation which activates the norm. Or conditions (1–3) could define a role *Bidder* and conditions (4) and (5) the conditions for the norm. Or conditions (1–4) could define the role *Highest Bidder* and condition (5) state when the norm applies. Finally all of conditions (1–5) could define the role *Successful Bidder*, and the norm would always apply to all agents playing this role.

Thus in designing any particular system, we will need to make a trade-off between how specific we make the roles, and the conditions needed to state when an agent in that role has a right or a duty. One can readily imagine a spectrum: at one extreme, the approach would be to use quite general roles, requiring extensive use of conditional norms; whereas, at the other extreme, the approach would use entirely specific roles, obviating the need for any conditional norms. Of course, there can be intermediate positions, as indicated in this example, and designers of any particular system will need to decide where along this spectrum their system will reside.

3.3 A method for dynamic assignment

Informed by the general discussion above, we now propose a method for the dynamic assignment of rights, responsibilities and sanctions to agents in MAS. We distinguish between *external* and *internal* agents in any open system. Internal agents are designed by the system designers, and are part of the apparatus of central control of the system. In particular, the system designers are able to fully specify their behaviour at design time, and can be assured that their behaviour will be fully legal and predictable. External agents, by contrast, are designed by other design teams (not the team designing the overall system), and typically only join the system at run-time. Neither their identities nor all their behaviours will be known to the system designers at design time. In a system using norm enforcement, external agents will retain some autonomy to violate the norms and regulations of the system. Internal agents have no such freedom of choice regarding the system norms. We draw this distinction in order to say that our methods of dynamic assignment are intended only to apply to external agents. For example, in an electronic auction system, the auction manager is an internal agent who follows the regulations of the system as defined at design time. But any external agent joining the system at run-time in order to act as a potential buyer or potential seller may choose to adhere to or to violate the system norms at run-time.

Our method is based on the use of conditional norms, which are are defined in the static normative KB at design time. This KB contains all rights and responsibilities associated with roles (including obligations, permissions, prohibitions or rights) and sanctions that can be applied to roles (including punishments, rewards and compensations) and the conditions which activate these various norms. We can see the normative KB as a set of rules of the form:

If preconditions A are true, then normative commands B must be applied.

For example, in an auction system, we could have a rule:

If a buyer has made the highest bid and the auction has ended, then that buyer is obliged to pay the price bid for the item.

In this method, roles are quite general and only the main roles are defined. For example, in the auction system the main roles of the system can be defined as Seller, Buyer and Auctioneer. In these rules, the preconditions of norms comprise the actions, events and status indicators that are preconditions for the activation of the norm. So whenever run-time occurrences cause the satisfaction of the preconditions of a norm of a role, the corresponding norm will be fired and assigned to any agent which is currently performing that role.

For dynamic assignment of rights, responsibilities and sanctions to agents using conditional norms, the following steps thus need to be executed:

(1) The normative KB is defined by the system designers. This KB contains all rights, responsibilities and sanctions applicable to the various roles, expressed as conditional norms. This is a design time task.
(2) The system assigns a main role to each agent immediately after joining the system, according to the initial action of the agent, e.g. at the beginning of an auction session, an agent may select to be either a buyer or a seller. In this method, the number of initial roles is limited and there are only a few specific actions that may lead to change the role of the agent. This is a run-time task.
(3) The system assigns a new role to an agent during system operation, precisely when some precondition is satisfied. If an agent executes any of the actions which may lead to a change in its role in the system, the system immediately assigns the new role to the agent. This is a run-time task.
(4) Immediately after occurrence of any action, environmental event or other run-time occureence, if the conditions of a norm, which will be related to a role, are satisfied, then the specific rights or responsibilities associated with that norm are assigned to each agent currently performing that role. This is a run-time task.

3.4 Designing the normative KB

The design of the normative KB is very important in the process of implementation of our approach. This KB is the source of all norms and regulations considered by the designer to influence the behaviour off the external agents using the system. These norms specify for each role which actions are obligatory, forbidden and permitted. In addition, the norms of the normative KB specify the responses of the normative MAS if the external agents deviate from the prescribed behaviour. This means that the enforcement norms are also defined in the normative KB.

Therefore, two important issues are the design of a normative language for specifying norms, and implementation of these norms in the normative KB. A detailed language for specifying norms is given in [7], and will not repeated here. For implementation, the generic syntax of a norm can be defined as a rule in the Jess language [11], and Jess was used as the inference engine of the KB. It would be quite possible to use other languages besides Jess for implementation of the normative KB.

4 Overlay architecture

In this section, we present a general architecture which may overlay a normative MAS in order to enable dynamic assignment of rights, responsibilities and sanctions to agents in a normative MAS. The architecture, shown in Figure 1, is intended to be implemented as a middleware tool available to any MAS seeking to use our method. In this section, we first describe the external components—linked to the central part of the architecture—and their connections to the main entities of our architecture. Later subsections describe the component entities and their communication processes.

4.1 External components

As can be seen at the top of the figure, the overlay architecture for dynamic assignment has interconnections to the external agents in a MAS. But these agents do not have direct contact with the

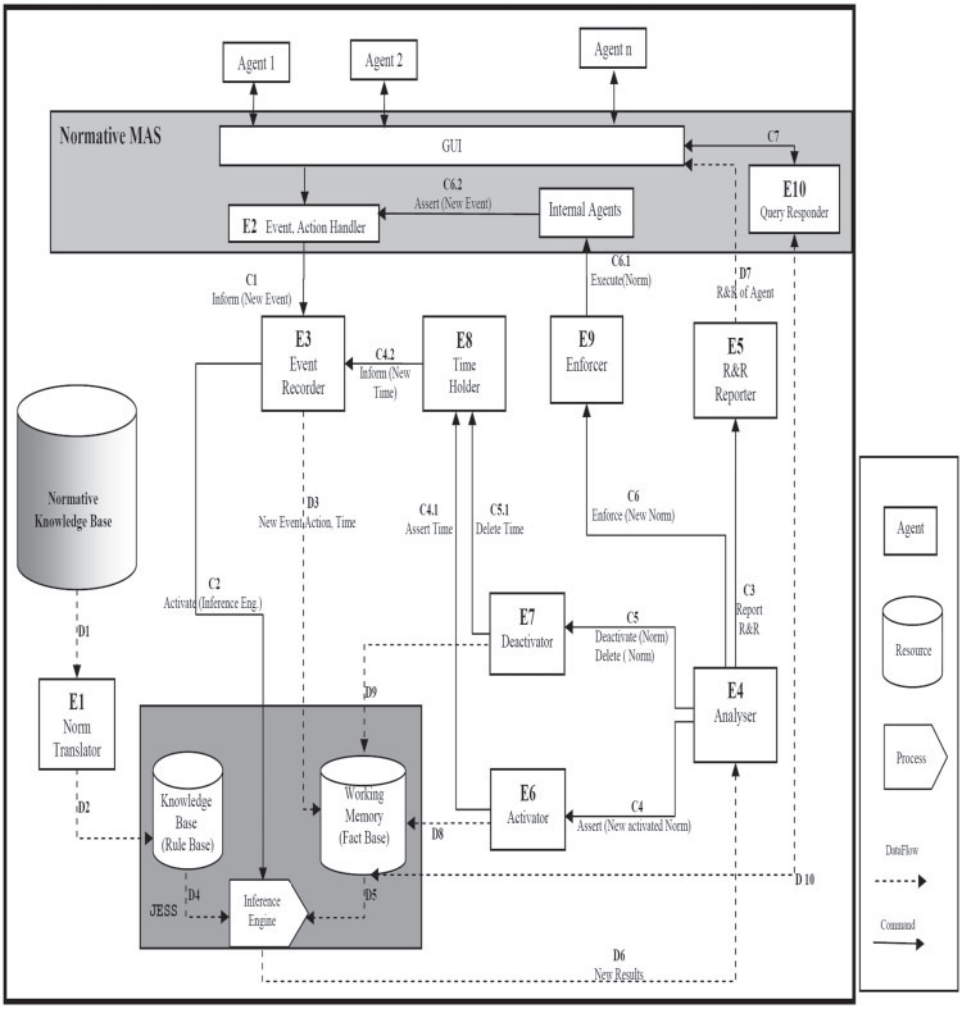

FIGURE 1. Overlay architecture for dynamic assignment of rights, responsibilities and sanctions to agents.

internal components of the architecture. We assume that the MAS has a component tasked with tracking the occurrence of all run-time events and actions of the MAS, a component called the Event/Action Handler; this component sends information on these occurrences to other internal components of the architecture.

On the left-hand side of the figure can be seen the normative KB which is the main normative resource of this overlay system. This KB is a static database of all domain-related norms including all rights, responsibilities and sanctions of roles, and all general rules for executing norms. We call this KB static, because we assume there are no changes to norms stored in it at run-time.

As stated above, for purposes of illustration we have assumed implementation using Jess for reasoning and inference. The three Jess components are shown in the shaded box at bottom, left; these components are a rule base, a fact base and an inference engine. In this structure, the rule base

is supplied by the normative KB (via a norm translator), and the fact base is supplied by the other internal components.

Internal Components in the overlay architecture are shown in the figure by boxes labelled with an upper-case E, followed by a number. The dotted arrows of the diagram labelled D represent data flows and the simple arrows labelled with C represent flow of control via communication. Arrows D1, D2, D4 and D5 indicate that data is transmitted from the respective source to the destination, while arrow D10 is a two-way arrow connecting Query Responder to Working Memory.

4.2 Overlay architecture components

We now describe the main entities and components in the overlay architecture, with the labels (E1, E2, etc) corresponding to the labels on the boxes in the Figure. We assume that external agents communicate with the overlay system by means of a component here labelled graphical user interface (*GUI*); however, this interface need be neither graphical nor human-understandable, since the agents in question will be software agents.

E1. Norm Translator: as we use a Jess rule engine to instantiate this architecture, the contents of the KB must be in the Jess language. The translation of the normative KB to the Jess language is the responsibility of the Norm Translator. The arrow labelled D1 represents the data link and provides the input for the Norm Translator from the normative KB, while arrow D2 sends the output from the Norm Translator to the Jess rule base.

E2. Event/Action Handler: this component is responsible for tracking and recording all run-time occurrences such as the environmental events, the entry and exit of agents, the actions of the MAS and important time thresholds (e.g. deadlines) being passed; each new event is reported to the Event Recorder (E3). Component E2 gets its inputs from the GUI and the environment of the MAS and its output is transferred to the Event Recorder (E3) via control link labelled C1.

E3. Event Recorder: this component is responsible for—first, asserting a new occurrence as a new fact to the Jess fact base; second, once the new occurrence is asserted as a fact, for the invocation of the Jess inference engine. The inputs of this entity come from Event Handler (E2), Time Holder (E8) and Enforcer (E9), via links C1, C4.2 and C6.2, respectively. The inputs contain a command for asserting a new event/action, a time or an internal parameter. After receiving inputs, this component produces the Jess fact format of its input to assert this fact to Jess. The output of this component is data in an acceptable format for the Jess fact base transferred via flow D3. The other output, C2, is a command for activating the Jess inference engine.

E4. Analyser: this component is responsible first for collecting the result of the Jess reasoning following each occurrence. Then this entity analyses Jess results using the Status slot of fired norms or enforcement norms. As described in [7], the status of norms include the five states, *ToBeActivated, Activated, Deactivated, Fulfilled* and *Violated*:

Based on the result of an inference, if a new obligation, permission, prohibition or right has been fired, this new right and responsibility transfers to the Reporter (E5) via flow C3. Any rules fired have to assert new facts in the fact base. The analyser requests the Activator (E6) via link C4 to assert these facts. For example, if *Activate(Agent Ali is forbidden to place a bid)* is inferred in an auction system, then the Analyser asks the Activator to activate this norm by putting the fired norm in fact base.

Based on the inference results, if a norm is fulfilled or violated and needs to be deactivated, then the norm should be retracted from the fact base. The Analyser asks the Deactivator (E7) to do this via link C5. Based on the inference results, if an internal activity in the MAS needs to be

executed, for example to apply a sanction, then the Analyser requests the Enforcer (E9) via link C6 to ask the internal agents to effect this. The internal activity can provide a change in the value of environmental variables (such as a feedback) or enforcing a punishment (such as disconnecting the offending agent). The input of this entity comes from the Inference engine via link D6 which contains the data of the new results following the Jess reasoning. The output transfers to the Reporter (E5), the Activator (E6), the Deactivator (E7) and the Enforcer (E9) through links C3, C4, C5 and C6, respectively.

E5. Rights and Responsibilities Reporter: this component is responsible for displaying the currently activated norms which have been assigned to the agents in the GUI. This entity receives all newly-fired norms and detects which norm is related to which agent to allocate its relevant rights and responsibilities. The input of this entity is a command for reporting rights and responsibilities, from the Analyser via C3, and the output of this entity is the data of assigned rights and responsibilities or sanctions to agents via D7, which will be displayed in the GUI. For example, if one of the newly-fired norms in an auction system is *Agent David is forbidden to place a bid*, the Reporter parses this statement to find which external agent the norm applies to and assigns the norm to that agent.

E6. Activator: this component is responsible for asserting the newly activated norms to the Jess fact base. In addition, if this norm contains a time-related parameter, the Activator is responsible for sending a message to the Time Holder (E8) to take note of that time threshold and to issue an alert when that time threshold is reached. The input of this entity is a command for asserting the activated norm via link C4, and the output of this entity is transferred via links D8 and C4.1, respectively containing data to the fact base and a command for asserting the time.

E7. Deactivator: this component is responsible for removing norms which have been fulfilled or which should be deactivated from fact base. In addition, if the norm had a time-related parameter which previously the Time Holder (E8) received, now the Deactivator should ask Time Holder (E8) to remove the norm from the list of important times. The input of this entity is a command for deactivating or deleting a norm via link C5, and the outputs of this entity transfer via links D9 and C5.1 which respectively show an access to fact base (for removing the norm) and a command for deleting the time of deactivated norm from the list of important times.

E8. Time Holder: at system start, the Time Holder makes a list of all important time thresholds expected to occur during run-time. Then whenever the current time passes each noted important threshold time, Time Holder sends the current time to the Event Recorder (E3) in order to assert the current time to the fact base. These important times are supplied by the Activator (E6) from the time notion of activated norms. However, it is possible that the Deactivator (E7) sends a message to remove this time from the list of important times, for example, because the norm is fulfilled before the deadline. The inputs of this entity are commands for asserting or deleting a time from the list of important threshold times, via links C4.1 or C5.1.

E9. Enforcer: this component is responsible for enforcing some internal actions. In fact, Enforcer interacts with internal agents and requests them to execute sanctions or rewards. Sometimes as a result of executing an enforcement norm, an environmental change occurs, reported by internal agents to the Event/Action Handler (E2). Then the Event/Action Handler (E2) sends a command to the Event Recorder (E3) for the assertion of this new event. The inputs of this entity are a command for the execution of a norm via link C6 and the output is one or more commands to the relevant Internal Agents via link C1.

E10. Query Responder: this component is responsible for answering the queries comes from GUI. This entity directly accesses the fact base. The input of this entity comes from GUI via link C7. This component has also access to the fact base via D10 to provide the answers to queries. The output will also transfer via link C7.

We separate the Reporter (E5) and the Query Responder (E10) because the responsibility of the Reporter is just to represent the rights and responsibilities of agents to the GUI at each moment of time. So the Reporter does not need access to the fact base directly, since that database contains much unrelated information. In contrast, the Query Responder needs to access the fact base to respond to various queries about norms.

4.3 Overlay architecture process

In this subsection, we provide a general description of the process of dynamic assignment of rights and responsibilities and sanctions to external agents, based on our proposed overlay architecture. We show precisely how such assignments occur, when, and by whom, by reference to Figure 1.

How Dynamic Assignment Occurs: Initially, this process needs a preparation stage before run-time. In this static stage, after identification or creation of system roles, the normative KB is created for those roles. Then if the language of this normative KB is different from the language of the Jess rule engine, the Norm Translator (E1) must be applied on the normative KB to provide the compatible version of the rule base for Jess. Otherwise, the Jess rule base is directly supplied with the original normative KB. In addition, we assume that all the connections between external agents and the multiagent system, between the components of the MAS and the components of our overlay system, and between the agents of the MAS and the Jess rule engine have been set.

At run-time, agents enter and join the system, they interact with the GUI and the GUI transfers their actions (as run-time occurrences) to the Event/Action Handler (E2) for handling the events or the actions. The Event/Action Handler (E2) then sends a message via link C1 to the Event Recorder (E3) to inform that an event has occurred. Next, the Event Recorder (E3) accesses the Jess fact base via link D3 to assert this event. Immediately after assertion, the Event Recorder (E3) asks via link C3 the Jess Inference Engine for reasoning. The Jess Inference Engine then reasons using the Jess fact base and the Jess rule base, after which the Analyser (E4) gets the results of reasoning from the Jess Inference Engine via link D6 to check which new norms, if any, have been fired by the recent occurrence. Jess results are a set of normative commands. Next, the Analyser (E4) parses the normative commands and, on the basis of the status of each normative command, makes decisions for the next stage as follows:

- If the normative command indicates an activation or deactivation, the Analyser (A4) sends the norm (or enforcement norm) to the Reporter (E5) via link C3 for reporting the rights and responsibilities.
- If the normative command is an activation, the Reporter (E5) parses the norm to specify which external agent this norm or enforcement norm concerns. Then, the norm (for rights and responsibilities) or the enforcement norm (for sanctions) needs to be assigned to the recognized external agent. Therefore, the Reporter (E5) provides the GUI format of the assigned norm and, using link D7, sends the data to display its specific frame in the GUI.
- If the normative command is a deactivation, the Reporter (E5) parses the norm to specify which external agent this norm or enforcement norm concerns. Then, the norm or enforcement norm needs to be removed from the specific frame of the external agent in the GUI. Therefore, Reporter (E5) will access the relevant frame via link D7 to remove the norm or enforcement norm.

At this stage, the dynamic assignment of rights and responsibilities, or sanctions, has been done for just one occurrence. However, for subsequent assignments of rights and responsibilities to agents for all occurrences of the system, the fact base will need to be updated (using the Activator,

Deactivator and Time Holder) and execution of internal enforcement actions undertaken (using Enforcer). Thus, the Analyser has the following additional tasks, based on the results of Jess reasoning:

- If the normative command indicates an activation, the Analyser (E4) sends a message via link C4 to the Activator (E6). The Activator then parses the norm. If the norm contains a time notion, it will ask the Time Holder (E8) to assert the time in its list of important times via link C4.1. In addition, the Activator (E6) will assert (update) the activated norm by accessing the Jess fact base via link D8.
- If the normative command indicates a deactivation, the Analyser (E4) sends a message via link C5 to the Deactivator (E7). The Deactivator then parses the norm. If the norm contains a time notion, it will ask, via link C5.1, the Time Holder (E8) to delete the time from its list of important times. In addition, the Deactivator will remove (update) the deactivated norm by accessing the Jess fact base via link D9.
- If the normative command indicates an execution, the Analyser (E4) sends a message via link C6 to the Enforcer (E9). Then, the Enforcer will send a message to the relevant Internal Agents to execute the enforcement norm via link C6.1. This message contains the statement that the agent should be punished along with the related enforcement code. If the execution of the enforcement norm leads to a run-time occurrence (an environmental event or events), such an occurrence should be reported to the Event/Action Handler (E2) for further action (repeating the cycle described above).

There is another case which leads to a repetition of the above cycle. We have mentioned that the Time Holder (E8) stores the important threshold times of run-time. Then, when the current time reaches any of these threshold times, the Time Holder will inform the Event Recorder (E3), via link C4.2, that the current time is an important theshold time. Therefore, this run-time occurrence—reaching an important theshold time—will be asserted to the Jess fact base through the Event Recorder (E3), and another process will be started.

In addition, if the system designers desire to facilitate responses to queries from external agents regarding the appropriate rights, responsibilities and sanctions, then the Query Responder (E10) may have direct access to the Jess fact base, via link D10, for answering such queries. Link C7 is a two-way path for receiving queries from the GUI and for responding to them via the GUI.

When a dynamic assignment occurs: Dynamic assignments may occur on the following occasions:

(1) *Entry and exit of agents:* on these occasions, an agent is assigned a role that is new to that agent. Thus, according to the rule base and fact base, dynamic assignment of rights and responsibilities occurs and the norms associated to the new role will be fired for the agent.
(2) *Occurrence of an Action:* when an action occurs, according to the rule base and the fact base, dynamic assignment of rights and responsibilities may take place. For example, an agent with the role of bidder in an auction system may make a bid, and thereby qualify for the right to receive an acknowledgement from the auctioneer within a certain time.
(3) *Occurrence of an Event:* when an event occurs, according to the rule base and the fact base, dynamic assignment of rights and responsibilities may take place. For example, an agent with the role of bidder in an auction system may have the right to purchase the item at the price bid if all other bids are lower than this price.
(4) *Reaching to an Important Time:* when the current time is an important threshold time, according to the rule base and the fact base, dynamic assignment of rights and responsibilities may

take place. For example, an auction in an auction system may have to be completed within a predefined time, such as within 1 hour of commencement.

Who assigns dynamic assignments: as the description in this subsection makes clear, all the component entities of the overlay structure contribute to the goal of providing dynamic assignment of rights and responsibilities to external agents. However, within this structure, the Reporter (E5) is the component which decides which activated norm or enforcement norm is associated with which external agent, and then asks the GUI to communicate the result of dynamic assignment to agents as these decisions are made. The Reporter also removes deactivated norms or enforcement norms from the GUI frames of external agents.

5 Related work and conclusions

In this article, we have outlined a novel and practical method for the dynamic assignment of rights, responsibilities and sanctions to agents in normative MAS at run-time. The method we have proposed is based on explicit representation of norms as conditional rules, where the preconditions for instantiation of a norm are various run-time occurrences—e.g. the execution or non-execution of some agent action, the occurrence of some environmental event, or the reaching of some time threshold. Our method has wide generality, and could be applied as an overlay to any multiagent system intended to have norm enforcement, i.e. allowing some agents the autonomy to adhere to or to violate the rules of the system. Our method also has great flexibility, allowing at one extreme few or no agent roles, each role having many conditional norms, or, at the other extreme, allowing many roles, with each role having only a small number of condition norms, or, indeed, any position between these two extremes.

Although we believe our work to be among the first which considers the dynamic, run-time assignment of rights, responsibilities and sanctions to agents in normative MAS, there has been earlier work on the dynamic assignment of roles to agents, since the notion of role is a key issue in the design and implementation of MAS [22]. For example, Odell *et al*. [26] consider the possible states or values which a fluent mapping agents to roles may take over time. Their key concern, however, is the design representation and programming implementation of these ideas using object-oriented techniques. Subsequently, Partsakoulakis and Vouros [27] presented a survey of the notion of roles in agent-oriented software engineering (AOSE) design methodologies and models, such as Gaia and AALAADIN, and in implemented MAS. For this survey, they consider various high-level attributes of roles, including their dynamic allocation to agents in an MAS. Most AOSE methodologies lack this feature, having only static design-time assignment, although the AALAADIN model of Ferber *et al*. [10] does allow dynamic creation and assignment of groups, which are viewed as collections of roles. Related work is that of Kim [18] which proposes a mechanism for dynamic role assignment in MAS where the agents are cooperative, as in a robot soccer team or in many computer games. In Kim's approach, roles are assigned at run-time according to emerging situations, based on rules predefined at design-time. Roles are viewed as situation-dependent collections of tasks, which are assigned to agents dynamically to enable more efficient operation of the overall system. None of these approaches or models consider roles as collections of inherent agent rights and responsibilities, as we do here. More recently, Balbiani *et al*. [4] have proposed a logical formalizm for specification of permissive rights to roles in automated control of access to resources. The formalizm separates static rights—those which remain constant throughout the life of a system—from dynamic properties—those which change in response to user actions during system operation. As in our approach, dynamic assignments occur upon realization of various triggers, but these are limited to the actions of a user

seeking access. Our approach has a wider class of triggers, namely the various types of run-time occurrences.

The work closest to the framework of this article is that of Dastani, Tinnemeier and Meyer [6], who recently proposed a programming language for normative multi-agent systems. Their language enables designers of an open multi-agent system to define and implement norms, along with mechanisms for monitoring and enforcement, and for the allocation of sanctions to agents in cases of non-compliance. The authors present the syntax and an operational semantics for the programming language, which enables ready implementation of an interpreter. The proposed language is at a higher level of abstraction than our framework, and may be more generic in respect of the types of norms considered. Perhaps because of the higher level of abstraction, it is much simpler than our framework. One difference is that, unlike our framework, agent roles are not considered explicitly, so that the task of specification of norms, rights, responsibilities and sanctions in the programming language of [6] may prove to be more cumbersome than in our framework, at least for some application domains.

As we discussed in Section 3.2, a system designer has a spectrum of alternatives in defining agent roles in a multiagent system and in allocating rights and responsibilities to these roles, from having many, very limited roles to having only a small number of very broad roles. In this article, we have presented a method for the dynamic assignment of rights and responsibilities to agents which is sufficiently generally to apply at any point along this spectrum. In related work [7], one of us has undertaken a comparison of two relatively extreme positions on the spectrum, and discussed the trade-offs involved for a system designer in choosing a position along this spectrum. In essence, the more roles are defined the greater the amount of time needed at design-time, and the less needed at run-time, to provide for dynamic assignment. In the extreme case, where agents may be assigned only role, designers need little time at design time, while agents, having a large and possibly conflicting set of rights and responsibilities for that role, will face more challenging, and thus slower, run-time decision-making. An interesting question for future work is whether there are particular multiagent application domains, or particular design-time or run-time circumstances, which favour particular points along this spectrum. Other potential future work includes evaluation of this framework, particularly in comparison with that of [6], with respect to software engineering criteria, such as implementation feasibility, scalability and integration with agent-oriented software engineering design methodologies and architectures.

Acknowledgments

This work formed part of a PhD undertaken by Dr Derakhshan at the University of Liverpool, UK, between 2004 and 2008, and she is grateful for financial support received from the Ministry of Science, Research and Technology of the Islamic Republic of Iran. We are grateful for comments received on earlier versions of this work from Katie Atkinson and Mark d'Inverno, and for the very constructive comments of the anonymous journal reviewers.

Funding

Ministry of Science, Research and Technology of the Islamic Republic of Iran.

References

[1] H. Aldewereld, A. García-Camino, F. Dignum, P. Noriega, J. A. Rodríguez-Aguilar, and C. Sierra. Operationalisation of norms for electronic institutions. In *Coordination, Organization,*

Institutions and Norms in agent systems II, Vol. 4386 of *Lecture Notes in Computer Science*, pp. 163–176. Springer, 2007.

[2] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *ACM Journal*, **49**, 672–713, 2002.

[3] J. L. Arcos, M. Esteva, P. Noriega, J. A. Rodriguez-Aguilar, and C. Sierra. Engineering open environments with electronic institutions. *Engineering Applications of Artificial Intelligence*, **18**, 191–204, 2005.

[4] P. Balbiani, Y. Chevalier, and M. El Houri. A logical approach to dynamic role-based access control. In *AIMSA*, Vol. 5253 of *Lecture Notes in Computer Science*, D. Dochev, M. Pistore, and P. Traverso, eds, pp. 194–208. Springer, 2008.

[5] E. Bulygin. On norms of competence. *Law and Philosophy*, **11**, 201–216, 1992.

[6] M. Dastani, N. Tinnemeier, and J.-J. Meyer. A programming language for normative multi-agent systems. In *Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, V. Dignum, ed., IGI Global, 2009.

[7] F. Derakhshan. *The Implementation of Dynamic Assignment of Rights, Responsibilities and Sanctions to External Agents in Normative Multiagent Systems*. PhD Thesis, Department of Computer Science, University of Liverpool, 2008.

[8] F. Derakhshan, P. McBurney, and T. Bench-Capon. Towards dynamic assignment of rights and responsibilities to agents. Report ULCS-08-008, Department of Computer Science, University of Liverpool, 2008.

[9] M. Esteva, J. A. Rodríguez-Aguilar, J. L. Arcos, C. Sierra, P. Noriega, and B. Rosell. Electronic institutions development environment. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pp. 1657–1658. IFAAMAS, 2008.

[10] J. Ferber and O. Gutknecht. AALAADIN: a meta-model for the analysis and design of organizations in multi-agent systems. In *Proceedings of the Third International Conference on MultiAgent Systems (ICMAS98)*, pp. 128–135. IEEE Computer Society, 1998.

[11] E. Friedman-Hill. *Jess in Action: Java Rule-Based Systems*. Manning, 2003.

[12] A. García-Camino, P. Noriega, and J. A. Rodríguez-Aguilar. Implementing norms in electronic institutions. In *Proceedings of the Fourth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, pp. 667–673. IFAAMAS, 2005.

[13] A. García-Camino, J. A. Rodríguez-Aguilar, C. Sierra, and W. W. Vasconcelos. Norm-oriented programming of electronic institutions: a rule-based approach. In *Coordination, Organization, Institutions and Norms in Agent Systems II*, Vol. 4386 of *Lecture Notes in Computer Science*, pp. 177–193. Springer, 2007.

[14] B. Gaudou, D. Longin, E. Lorini, and L. Tummolini. Anchoring institutions in agents' attitudes: towards a logical framework for autonomous MAS. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, L. Padgham and D. C. Parkes, eds, pp. 728–735. IFAAMAS, 2008.

[15] H. L. A. Hart. *The Concept of Law*. Clarendon Press, 1992. (First published 1961).

[16] H. Herrestad and C. Krogh. Obligations directed from bearers to counterparts. In *Proceedings of the Fifth International Conference on Artificial Intelligence and Law (ICAIL 1995)*, pp. 210–218. ACM, 1995.

[17] A. Jones and M. J. Sergot. A formal characterization of institutionalised power. *Logic Journal of the IGPL*, **4**, 429–445, 1996.

[18] I. C. Kim. Dynamic role assignment for multi-agent cooperation. In *Computer and Information Sciences - ISCIS 2006*, Vol. 4263 of *Lecture Notes in Computer Science*, pp. 221–229. Springer, 2006.

[19] R. W. C. Kralingen, P. R. S. Visser, T. J. M. Bench-Capon, and J. Herik. A principled methodology to developing legal knowledge systems. *International Journal of Human-Computer Studies*, **51**, 1127–1154, 1999.
[20] D. K. Lewis. *Convention: A Philosophical Study*. Harvard University Press, 1969.
[21] E. Lorini, D. Longin, B. Gaudou, and A. Herzig. The logic of acceptance: grounding institutions on agents' attitudes. *Journal of Logic and Computation*, **19**, 901–940, 2009.
[22] M. Luck, P. McBurney, O. Shehory, and S. Willmott. *Agent Technology: Computing as Interaction. A Roadmap for Agent Based Computing*. AgentLink III, the European Co-ordination Action for Agent-Based Computing, 2005.
[23] S. Munroe, T. Miller, R. Belecheanu, M. Pechoucek, P. McBurney, and M. Luck. Crossing the agent technology chasm: lessons, experiences and challenges in commercial applications of agents. *Knowledge Engineering Review*, **21**, 345–392, 2006.
[24] T. J. Norman and C. Reed. Delegation and responsibility. In *Intelligent Agents VII*, Vol. 1986 of *Lecture Notes in Artificial Intelligence*, Y. Lesperance and C. Castelfranchi, eds, pp. 136–149. Springer, 2001.
[25] T. J. Norman and C. Reed. A logic of delegation. *Artificial Intelligence*, **174**, 51–71, 2010.
[26] J. Odell, H. V. D. Parunak, S. Brueckner, and J. A. Sauter. Changing roles: dynamic role assignment. *Journal of Object Technology*, **2**, 77–86, 2003.
[27] I. Partsakoulakis and G. Vouros. Roles in MAS: Managing the complexity of tasks and environments. In *Multi-Agent Systems: An Application Science*, T. Wagner, ed., pp. 133–154. Kluwer Academic, 2004.
[28] J. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969.
[29] M. P. Singh. An ontology for commitments in multiagent systems: toward a unification of normative concepts. *Artificial Intelligence and Law*, **7**, 97–113, 1999.
[30] G. Therborn. Back to norms! On the scope and dynamics of norms and normative action. *Current Sociology*, **50**, 863–880, 2002.
[31] C. van Aart, K. Van Marcke, R. Pels, and J. Smulders. International insurance traffic with software agents. In *Proceedings of the Fifteenth European Conference on Artificial Intelligence (ECAI 2002)*, F. van Harmelen, ed., pp. 623–627. Lyon, 2002.
[32] J. Vázquez-Salceda, H. Aldewereld, and F. Dignum. Norms in multiagent systems: from theory to practice. *International Journal of Computer Systems Science and Engineering*, **20**, 225–236, 2005.

Received June 7 2010