The Norm Implementation Problem in Normative Multi-Agent Systems

Davide Grossi[1], Dov Gabbay[2,3], Leendert van der Torre[3]

[1]ILLC, University of Amsterdam
d.grossi@uva.nl
[2]King's College London
dov.gabbay@kcl.ac.uk
[3]ICR, University of Luxembourg
leendert@vandertorre.com

Abstract. The norm implementation problem consists in how to see to it that the agents in a system comply with the norms specified for that system by the system designer. It is part of the more general problem of how to synthesize or create norms for multi-agent systems, by, for example, highlighting the choice between regimentation and enforcement, or the punishment associated with a norm violation. In this paper we discuss how various ways to implement norms in a multi-agent system can be distinguished in a formal game-theoretic framework. In particular, we show how different types of norm implementation can all be uniformly specified and verified as types of transformations of extensive games. We introduce the notion of retarded preconditions to implement norms, and we illustrate the framework and the various ways to implement norms in the blocks world environment.

Keywords. Normative systems, multi-agent systems, extensive games, logic.

1 Introduction

Normative multi-agent systems (NMAS) [15] study the specification, design, and programming of systems of agents by means of systems of norms. Norms allow for the explicit specification of the standards of behavior the agents in the systems are supposed to comply with. Once such a set of norms is settled, the question arises of how to organize the agents' interactions in the system, in such a way that those norms do not remain—so to say—dead letter, but they are actually followed by the agents. Designing a NMAS does not only mean to state a number of standards of behavior in the form of a set of norms, but also to organize the system in such a way that those standards of behavior are met by the agents participating in the system. In other words, norm creation [11] distinguishes between the creation of the obligation and norm implementation, because these two problems have different concerns. On the one hand the creation of the obligation says how the ideal can be reached, and the creation of the

sanction says how agents can be motivated to comply with the norms such that the ideal will (probably) be reached. The paper moves the first steps towards a formal understanding of the norm implementation problem, defined as follows.

> *The norm implementation problem.* How to make agents comply with a set of norms in a system?

In this paper we introduce a formal framework that can represent various solutions to the norm implementation problem, which can be used to analyze them, or to make a choice among them. For example, in some cases a norm cannot be regimented, such as the norm to return books to the library within two weeks, but in other cases there is the choice between regimentation and enforcement. It is often assumed that regimenting norms makes the system more predictable, since agents cannot violate the norms, but as a consequence it also makes the system less flexible and less efficient. Conceptually, regimentation is easier than enforcement, and since agents are bounded reasoners who can make mistakes, regimentation is often favored by policy makers. However, policy makers are bounded reasoners too, who have to make norms in uncertain circumstances, and therefore most people prefer enforcement over regimentation—at least, when the legal system is reliable. As another example, it is often assumed that very high punishments make the system less efficient than lower ones, due to the lack of incentives for agents once they have violated a norm. Such assumptions are rarely studied formally.

Although ideas about norm implementation can be found scattered over, in particular, the multi-agent systems literature (for instance, in OperA [27], $\mathcal{M}$oise$^+$ [41], AMELI [29], J-Moise+ [42], and in programming languages for multi-agent system programs [26]), they have not yet been presented in a systematic and uniform way. One of the aims of the paper is to do so, providing a formal overarching framework within which it becomes possible to place and compare existing contributions. So the first requirement on a framework for norm implementation is that it can represent existing widely discussed norm implementation methods such as regimentation and enforcement via sanctioning. Moreover, a second requirement is that such a framework can also represent and reason about new ways of norm implementation, such as changing the existing norms or the method, which we will study in detail, of retarded preconditions. A formal framework may even suggest new ways to implement norms, not discussed before. Our research problem therefore breaks down into the following sub-questions:

1. How is the specification and verification of norm implementation methods related to general specification and verification of multiagent systems?
2. Which formal model can we use to specify and verify norm implementation, and more generally study the norm implementation problem?
3. How to model regimentation in the general formal model of norm implementation?
4. How to model enforcing in the general formal model of norm implementation?

5. How to model norm change in the general formal model of norm implementation?

The specification of normative multi-agent systems often considers a set of agents and a set of norms or organization, which can be specified and verified independently. However, when the set of norms is not designed off-line, but created dynamically at run-time, then this approach does not work. Instead, one can only specify the way in which norms are implemented in a system.

The perspective assumed for the formal framework is based on formal logic and the primary aim of the paper is to present a simple class of logical models, and of transformations on them, as salient representations of the implementation problem. Moreover, we use a game-theoretic approach. We use the simplest approach possible, so we can focus on one same framework for many kinds of norm implementation, and are not lost in technical details about individual approaches. As in classical game theory, our actions are abstract and we do not consider issues like causality. We consider only perfect information games, and we thus do not consider the problem of how norms are distributed and communicated to a society. Moreover, we do not consider concurrent actions of the agents in the composition of actions in plans, although this feature could easily be added. However, we do represent the order of actions, that is, we use extensive form games, because we need to do so to distinguish some of the norm implementation methods, and thus we do not use the more abstract strategic form used in most related work (such as Tennenholtz and Shoham's artificial social systems [62]). We do not go into details of solution concepts of game theory, and we thus basically use a kind of state automata or process models. Within a game-theoretic approach, we need to represent four things: the game without the implemented norm, the game together with the implemented norm, the procedure to go from the former to the latter, and the compliance criterion stating the conditions when norms are fulfilled.

We test our model also by introducing the notion of implementation via retarded preconditions. For example, assume that in the tax regime of a country, for people who leave the country there is a period of three years after which it is checked whether someone has really left the country. In this example, the precondition is checked only after three years, and if the person has returned to the country, the consequences of leaving the country are retracted. Likewise, with actions with nondeterministic effects, we can say that the precondition depends on the effect. For example, if there are no concurrent updates in the database, then the update will be accepted, otherwise it will be rolled back. In the blocks world, which will be used as running example throughout the paper, assume that a block may not be put on another block if it stays there for three minutes. If it stayed there for three minutes, then we can undo the action of putting the block on top of the other one (alternatively, one can sanction it, of course). Retarded preconditions offer more flexibility than simple regimentation. For example, consider the norm that it is forbidden to throw 6 on a dice. With retarded preconditions, we can throw the dice and do a roll-back when 6 appears. Without retarded preconditions, the only way to regiment it, is to

forbid throwing the dice. We distinguish norm regimentation from automatic enforcement and enforcement agents, assuming actions can be taken only if preconditions hold. Some of such actions are forbidden, so all actions in order to be taken must satisfy the precondition. For regimentation we consider violation conditions as retarded preconditions of actions. In this action model, assumed actions can be taken in some cases even though the preconditions do not hold. When a violation occurs, i.e., when the retarded precondition does not hold, the various strategies to implement norms follow as a consequence.

We illustrate the framework and the various ways to implement norms in the blocks world environment, because the well-known planning environment explains the use of normative reasoning and the challenges of norm implementation for a large AI audience. There are many variants of the blocks world around, we use a relatively simple one with deterministic actions and without concurrent actions. An alternative well-known example we could have chosen is the Wumpus world from Russel and Norvig's textbook [60].

We assume that all norms and their implementations are known once they are created, and we thus do not study the norm distribution problem. Moreover, we assume that everyone accepts the existence of a new norm, even when he does not comply with it. Thus, we do not consider the norm acceptance problem. We do not consider cognitive aspects of agents, and we thus do not consider the bridge between our framework for MAS and existing BDI frameworks for cognitive agents (see, e.g., [14]).

The paper follows the research questions and proceeds as follows. In Section 2 we give a short introduction in the use of normative systems in computer science in general, and specification and verification of normative multi-agent systems in particular. In Section 3 we start with the game-theoretic framework for norm implementation and a logic for representing extensive games, and we introduce a running example. Sections 4, 5, 6, and 7 provide formal semantics to the four implementation strategies of regimentation, enforcement, enforcers, and, respectively, normative change. In presenting such semantics due care will be taken to relate our framework to existing literature showing how the framework is general enough to categorize, at a higher abstraction level, the various contributions available in the literature. The findings of each section is illustrated by means of the running example. In Section 8 related work at the intersection of norms and multi-agent systems is discussed. Conclusions follow in Section 9.

2 Normative multi-agent systems

In this section we first give a short summary of the main issues in using normative systems in computer science, and thereafter we discuss the specification and verification of normative multi-agent systems.

2.1 Normative systems in computer science

The survey of the role of normative systems in computer science in this section is taken from [7]. For a discussion on philosophical foundations for normative multi-agent systems, see [8,37].

There is an increasing interest in normative systems in the computer science community, due to the observation five years ago in the so-called AgentLink Roadmap [51, Fig. 7.1], a consensus document on the future of multi-agent systems research, that norms must be introduced in agent technology in the medium term (i.e., now!) for infrastructure for open communities, reasoning in open environments and trust and reputation. The first definition of a normative multi-agent system emerged after two days of discussion at the first workshop on normative multi-agent systems NorMAS held in 2005 as a symposium of the Artificial Intelligence and Simulation of Behaviour convention (AISB) in Hatfield, United Kingdom:

The normchange definition. "A normative multi-agent system is a multi-agent system together with normative systems in which agents on the one hand can decide whether to follow the explicitly represented norms, and on the other the normative systems specify how and in which extent the agents can modify the norms" [15].

A distinction has been made between systems in which norms must be explicitly represented in the system (the 'strong' interpretation) or that norms must be explicitly represented in the system specification (the 'weak' interpretation). The motivation for the strong interpretation of the explicit representation is to prevent a too general notion of norms. Any requirement can be seen as a norm the system has to comply with; but why should we do so? Calling every requirement a norm makes the concept empty and useless. The weak interpretation is used to study the following two important problems in normative multi-agent systems.

Norm compliance. How to decide whether systems or organizations comply with relevant laws and regulations? For example, is a hospital organized according to medical regulations? Does a bank comply with Basel 2 regulations?

Norm implementation. How can we design a system such that it complies with a given set of norms? For example, how to design an auction such that agents cannot cooperate?

Norms are often seen as a kind of (soft) constraints that deserve special analysis. Examples of issues which have been analyzed for norms but to a less degree for other kinds of constraints are ways to deal with violations, representation of permissive norms, the evolution of norms over time (in deontic logic), the relation between the cognitive abilities of agents and the global properties of norms, how agents can acquire norms, how agents can violate norms, how an agent can be autonomous [24] (in normative agent architectures and decision

making), how norms are created by a legislator, emerge spontaneously or are negotiated among the agents, how norms are enforced, how constitutive norms are used to describe institutions, how norms are related to other social and legal concepts, how norms structure organizations, how norms coordinate groups and societies, how contracts are related to contract frames and contract law, how legal courts are related, and how normative systems interact?

Norms can be changed by the agents or the system, which distinguishes this definition of normative multi-agent system from the common framework used in the Deontic Logic in Computer Science (or ΔEON) community, and led to the identification of this definition as the "normchange" definition of normative multi-agent systems. For example, a norm can be made by an agent, as legislators do in a legal system, or there can be an algorithm that observes agent behavior, and suggests a norm when it observes a pattern. The agents can vote on the acceptance of the norm. Likewise, if the system observes that a norm is often violated, then apparently the norm does not work as desired, and it undermines the trust of the agents in the normative system, so the system can suggest that the agents can vote whether to retract or change the norm.

After four days of discussion, the participants of the second workshop on normative multi-agent systems NorMAS held as Dagstuhl Seminar 07122 in 2007 agreed to the following consensus definition:

The mechanism design definition. "A normative multi-agent system is a multi-agent system organized by means of mechanisms to represent, communicate, distribute, detect, create, modify, and enforce norms, and mechanisms to deliberate about norms and detect norm violation and fulfilment." [17]

According to Boella *et al.*, "the emphasis has shifted from representation issues to the mechanisms used by agents to coordinate themselves, and in general to organize the multi-agent system. Norms are communicated, for example, since agents in open systems can join a multi-agent system whose norms are not known. Norms are distributed among agents, for example, since when new norms emerge the agent could find a new coalition to achieve its goals. Norm violations and norm compliance are detected, for example, since spontaneous emergence norms of among agents implies that norm enforcement cannot be delegated to the multi-agent infrastructure." [17] They refer to game theory in a very liberal sense, not only to classical game theory studied in economics, which has been criticized for its ideality assumptions. Of particular interest are alternatives taking the limited or bounded rationality of decision makers into account.

Games can explain that norms should satisfy various properties to be effective as a mechanism to obtain desirable behavior. For example, the system should not sanction without reason, as the norms would loose their force to motivate agents. Moreover, sanctions should not be too low, but they also should not be too high. Otherwise, once a norm is violated, there is no way to prevent further norm violations.

Games can explain also the role of various kinds of norms in a system. For example, assume that norms are added to the system one after the other and this

operation is performed by different authorities at different levels of the hierarchy. Lewis "master and slave" game [48] shows that the notion of permission alone is not enough to build a normative system, because only obligations divide the possible actions into two categories or spheres: the sphere of prohibited actions and the sphere of permitted (i.e., not forbidden) actions or "the sphere of permissibility". More importantly, Bulygin [22] explains why permissive norms are needed in normative systems using his "Rex, Minister and Subject" game. "Suppose that Rex, tired of governing alone, decides one day to appoint a Minister and to endow him with legislative power. [...] an action commanded by Minister becomes as obligatory as if it would have been commanded by Rex. But Minister has no competence to alter the commands and permissions given by Rex." If Rex permits hunting on Saturday and then Minister prohibits it for the whole week, its prohibition on Saturday remains with no effect.

As another example, Boella and van der Torre's game theoretic approach to normative systems [16] studies the following kind of normative games.

Violation games: interacting with normative systems, obligation mechanism, with applications in trust, fraud and deception.

Institutionalized games: counts-as mechanism, with applications in distributed systems, grid, p2p, virtual communities.

Negotiation games: MAS interaction in a normative system, norm creation action mechanism, with applications in electronic commerce and contracting.

Norm creation games: multi-agent system structure of a normative system, permission mechanism, with applications in legal theory.

Control games: interaction among normative systems, nested norms mechanism, with applications in security and secure knowledge management systems.

Norms are not only seen as the mechanism to regulate behavior of the system, but they are often also part of a larger institution. This raises the question what precisely the role of norms is in such an organization. Norms are rules used to guide, control, or regulate desired system behavior. However, this is not unproblematic, since norms can be violated, and behavior of agents may change in unexpected ways when norms are introduced due to self organization. Norms can also be seen as one of the possible incentives to motivate agents, which brings us again back to economics. The fact that norms can be used as a mechanism to obtain desirable system behavior, i.e., that norms can be used as incentives for agents, implies that in some circumstances economic incentives are not sufficient to obtain such behavior. For example, in a widely discussed example of the so-called centipede game, there is a pile of thousand pennies, and two agents can in turn either take one or two pennies. If an agent takes one then the other agent takes turn, if it takes two then the game ends. A backward induction argument implies that it is rational only to take two at the first turn. Norms and trust have been discussed to analyze this behavior, see [40] for a discussion.

A rather different role of norms is to organize systems. To manage properly complex systems like multi-agent systems, it is necessary that they have a mod-

ular design. While in traditional software systems, modularity is addressed via the notions of class and object, in multi-agent systems the notion of organization is borrowed from the ontology of social systems. Organizing a multi-agent system allows to decompose it and defining different levels of abstraction when designing it. Norms are another answer to the question of how to model organizations as first class citizens in multi-agent systems. Norms are not usually addressed to individual agents, but rather they are addressed to roles played by agents [12]. In this way, norms from a mechanism to obtain the behavior of agents, also become a mechanism to create the organizational structure of multi-agent systems. The aim of an organizational structure is to coordinate the behavior of agents so to perform complex tasks which cannot be done by individual agents. In organizing a system all types of norms are necessary, in particular, constitutive norms, which are used to assign powers to agents playing roles inside the organization. Such powers allow to give commands to other agents, make formal communications and to restructure the organization itself, for example, by managing the assignment of agents to roles. Moreover, normative systems allow to model also the structure of an organization and not only the interdependencies among the agents of an organization. Roles are played by other agents, real agents (human or software) who have to act as expected by their role. Each of these elements can be seen as an institution in a normative system, where legal institutions are defined by Ruiter [59] as "systems of [regulative and constitutive] rules that provide frameworks for social action within larger rule-governed settings". They are "relatively independent institutional legal orders within the comprehensive legal orders".

The second NorMAS workshop identified a trend towards a more dynamic interactionist view: "This shift of interest marks the passage of focus from the more static legalistic view of norms (where power structures are fixed) to the more dynamic interactionist view of norms (where agent interaction is the base for norm related regulation)." This ties in to what Strauss [64] called "negotiated order", Goffman's [33] view on institutions, and Giddens' [32] structuration theory. See [17] for a further discussion.

2.2 Specification and verification of normative multi-agent systems

The motivation of our work is to provide an answer to the more general issue of finding a logical formalism that could play for programming NMAS the role that BDI logics (e.g. [57]) have played for the programming of single agents. Such an issue was recognized as central for the NMAS community during the NorMAS'07 Datstuhl Seminar [16], and it was raised in the following incisive form:

$$\text{BDI} : \text{Agent Programming} = ? : \text{NMAS Programming}.$$

This equation represents two issues. First, it raises the question about which concepts should be used for programming NMAS, given that cognitive concepts like beliefs, desires and intentions are used to program individual agents. There

is some consensus that instead of cognitive concepts, for normative multi-agent systems social and organizational concepts are needed, such as trust, norms and roles. Second, from a logical perspective, it raises the question which logical languages used for specification and verification can be used for NMAS, like BDI-CTL is used for single agents. Thus far, only partial answers have been given to this question. For example, deontic logic can be used to represent norms, but it cannot be used to say how agents make decisions in normative systems, and about the multi-agent structure of normative systems.

In the traditional framework of artificial social systems, norms are designed off-line [62]. Thus, a norm is part of the specification of the multi-agent system, and the normative multi-agent system can be specified and verified using traditional techniques. For example, since BDI-CTL [23] is used as a formal specification and verification language for agent programming, and it has been extended with deontic concepts such as obligations and permissions, called BOID-CTL [20,21]. Such a logic is simply a modal combination of an agent logic and a modal deontic logic. One drawback of this approach is that the norms are not represented explicitly, see Section 8. However, a more fundamental problem with this approach for the specification and verification of normative multi-agent systems is that it is difficult to generalize this approach to the case where norms are created or synthesized at run-time.

The main challenge of specification and verification of normative multiagent systems is the specification and verification of norm change, and in particular the specification and verification of norm creation. Norm creation distinguishes between the creation of the obligation or prohibition, and the creation of the associated sanction. For example, the obligation may be to return books to the library within three weeks, and the sanction associated with its violation is that a penalty has to be paid, and no other books can be borrowed. The creation of the obligation is often called the norm design or synthesis problem [62], and the creation of the sanction is an example of what we call the norm implementation problem. Thus, in the library example, the norm implementation problem is that given that we want people to return their books within three weeks, how can we build a system such that they will actually do so? However, introducing sanctions is not the only way to implement norms. In other cases, the norm can be regimented, or instead of penalties, rewards can be introduced.

An alternative motivation to break down the specification of a normative multi-agent system is common in computer science: divide and conquer. We distinguish the specification and verification of normative multi-agent systems in three steps: the specification and verification of the agents, the specification and verification of the normative system, and the specification and verification of combining these two systems: the norm implementation problem. This reflects a common ontology of normative multi-agent systems. For example, Figure 1 shows the ontology of Boella et al [18] containing a number of concepts related to each other. They divide their ontology in three submodels: the agent model, the institutional model, and the role assignment model, as shown in Figure 1. Roughly, an institution is a structure of social order and cooperation govern-

ing the behavior of a set of individuals. Institutions are identified with a social purpose and permanence, with the enforcing of rules governing cooperative human behavior. The figure visualizes the three submodels which group the concepts of their ontology.

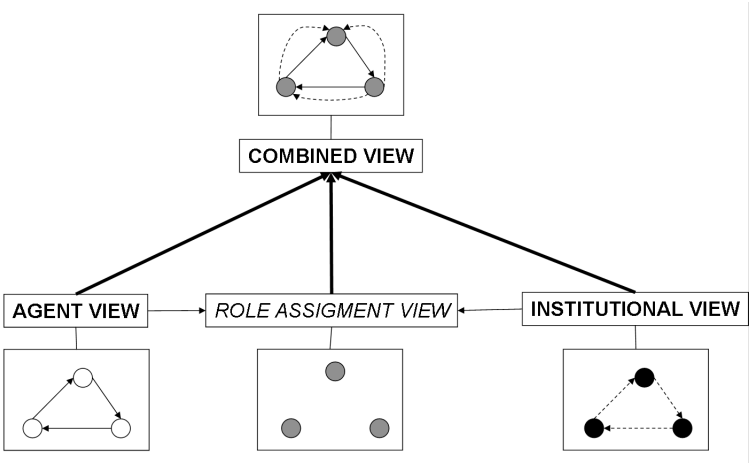

Fig. 1. The conceptual metamodel [18].

As Boella *et al.* observe, such a decomposition is common in organizational theory, because an organization can be designed without having to take into account the agents that will play a role in it. For example if a node with the role of simple user becomes a VO administrator, then this remains transparent for the organizational model. Likewise, agents can be developed without knowing in advance in which institution they will play a role. As shown in Figure 1, the agent view is composed by concepts like agent, goal and skill or ability and they are represented by means of a social dependence networks in which nodes are the agents and the edges are the representation of goal-based dependencies. The institutional view, instead, is composed by the notion of role and its institutional goals, skills and facts. As for the agent view, also the institutional one is represented by means of a social institutional dependence network representing the norm-based dependency relations among roles. The role assignment view associates to each agent the roles it plays, depending on the organization in which the agent is playing. All these notions are unified in the combined view where the dependence network represents at the same time both goal-based dependencies and norm-based ones connecting the agents playing roles.

The norm implementation problem combines the specification and verification of agents and norms, analogous to the role assignment problem combines these two specifications. However, it does not consider organizational issue of role assignment, but the question how to ensure that agents do comply with the norms. The decomposition in the role assignment problem is based on the

rationale that organizations must be designed independently of the agents that will play a role in it. The decomposition for the norm implementation problem is based on the rationale that in specifying a normative system, it makes sense to first specify the sub-ideal states the system should avoid, and thereafter how to ensure that the system avoids these sub-ideal states. If one norm implementation method does not work, then it can be replaced by another one, without changing the underlying norms.

2.3 Assumptions of norm implementation

Summarizing, the norm implementation problem is the part of the more general problem of norm creation which lends itself to specification and verification, since it focusses on the well-defined choice between regimentation and enforcement, or the punishment associated with a norm violation. For example, whether it is hard to give general guidelines for the violation states, since they can be defined by the agents at run-time, it is more straightforward to specify how these violation states must be avoided.

The assumption underlying our research problem is that the norm implementation problem can be studied in isolation. We thus disagree with the common idea that norm implementation can be studied only together with the norm design problem, in the context of norm creation. For example, when a system designer has to choose among various kinds of norms, at the same time he has to take into account how the norm can be implemented. If a norm is chosen which cannot be implemented, such that it will not be complied with, then the norm may even be counterproductive, undermining the belief or faith into the normative system (in particular, this holds for legal systems). Though we agree that a choice among norms also has to take the available implementations into account, we believe that this is not an argument against studying norm implementation in isolation.

3 Formal framework and running example

The present section sets the stage of our formal investigations.

3.1 Norms and logic

The formal representation of norms by means of logic has a long-standing history. In the present paper we assume a very simple perspective based on [2, 47, 53] representing the content of norms as labeling of a transition systems in legal and illegal states, which we will call *violation states*. In this view, the content of a normative system can be represented by a set of statements of the form:

$$\mathtt{pre} \rightarrow [\mathbf{a}]\mathtt{viol} \tag{1}$$

that is, under the conditions expressed in `pre`, the execution of action **a** necessarily leads to a violation state. Such statements can be viewed as constraints

on the labeling of transition systems. Restating Formula (1), all states which are labelled `pre` are states such that by executing an **a**-transition, states which are labelled `viol` are always reached.[1]

It follows that a set of formulae as Formula (1) defines a set of labelled transition systems (i.e., the set of transition systems satisfying the labeling constraints stated in the formulae), and such a set of transition systems can be viewed as representing the content of the normative system specified by those formulae.

Now, within a set of transition systems modeling a set of labeling constraints, transition systems may make violation states possibly reachable by transitions in the systems, and others possibly not. So, from a formal semantics perspective, we can think of the implementation problem as the problem of selecting those transition systems which:

1. Model a given normative system specification in terms of labeling constraints like Formula (1);
2. Make some violation states unreachable within the transition system, hence *regimenting* [46] the corresponding norms;
3. Make other violation states reachable but, at the same time, disincentivizing the agents to execute the transitions leading to those states, for instance by triggering appropriate systems reactions such as sanctioning, thus *enforcing* the corresponding norms [35].

To sum up, normative systems can be studied as sets of labeling constraints on the systems' transitions generated by agents' interaction, and the implementation problem amounts to designing the NMAS according to those transition systems which, on the one hand, model the labeling constraints and, on the other hand, make the agents' access to violation states either impossible (regimentation), or irrational (enforcement). What we mean by the term "irrational" is precisely what is studied by game theory [56], because due to punishments for norm violations the agent is motivated to fulfill the norm. Of course, this does not exclude the possibility that in some circumstances an agent may ignore this incentive and violate the norm. On the contrary, this is one of the reasons why sometimes enforcement is preferred to regimentation, because the creator of the norm cannot foresee all possible circumstances, and it is left to the rational agent to accommodate the local circumstances. The next section moves to the fundamental role that —we think— game theory can play for the analysis of the norm implementation problem.

3.2 Norm implementation and games

In a social setting, like the one presupposed by NMAS, action essentially means interaction. Agents' actions have repercussions on other agents which react accordingly. Norm enforcement takes care that agents' actions leading to violation states happen to be successfully deterred, either by a direct system reaction or,

[1] The reader is referred to [5] for more details on the logical study of labelled transition systems.

as we will see, by means of other agents' actions. The readily available formal framework to investigate this type of social interaction is, needless to say, game theory. The present paper uses the term implementation in the technical sense of implementation theory, i.e., that branch of game theory which, together with mechanism design [43–45, 52], is concerned with the design of the interaction rules—the "rules of the game" [54] or *mechanisms*—to be put into place in a society of autonomous self-interested agents in order to guarantee that the interactions in the society always result in outcomes which, from the point of view of the society as a whole (or from the point of view of a social designer), are considered most desirable (e.g., outcomes in which social welfare is realized).[2]

In this paper we are going to work with games in extensive forms [56]. Games in extensive form have recently obtained wide attention as suitable tools for the representation of social processes [3]. However, the key advantage for us of choosing games in extensive form is that such games are nothing but tree-like transition systems. This allows us to directly apply the logic-based representation of norms exposed in Section 3.1, thus obtaining a uniform formal background for talking about both norms and games and, hence, for formulating the norm implementation problem in an exact fashion. To ease such exact formulation, we will make use of a simple running example.

3.3 Running example: ruling the Blocks World

We assume a multi-agent variant of the blocks world, where agents cannot do concurrent actions (so we do not consider the issue of lifting a block simultaneously). Therefore we assume that the agents have to take actions in turn.

In the standard blocks world scenario [60], the pre- and postcondition specification of the action $\mathtt{move}(\mathsf{a},\mathsf{b},\mathsf{c})$ ("move block a from the top of b to the top of block c") runs as follows:

$$(\mathtt{on}(\mathsf{b},\mathsf{a}) \wedge \mathtt{clear}(\mathsf{c}) \wedge \mathtt{clear}(\mathsf{b}) \wedge \mathtt{turn}(i)) \leftrightarrow \langle \mathtt{move}(\mathsf{b},\mathsf{a},\mathsf{c})(i)\rangle \top \tag{2}$$

$$(\mathtt{on}(\mathsf{b},\mathsf{a}) \wedge \mathtt{clear}(\mathsf{c}) \wedge \mathtt{clear}(\mathsf{b}) \wedge \mathtt{turn}(i)) \rightarrow [\mathtt{move}(\mathsf{b},\mathsf{a},\mathsf{c})(i)]((\mathtt{on}(\mathsf{b},\mathsf{c}) \wedge \mathtt{clear}(\mathsf{b}) \wedge \mathtt{clear}(\mathsf{a})) \tag{3}$$

that is to say: the robotic arm i can execute action $\mathtt{move}(\mathsf{b},\mathsf{a},\mathsf{c})$ iff it is the case that both blocks b and c are clear, and it is its 'turn' to move;and the effect of such action is that block b ends up to the top of block c while block a becomes clear. By permutation of the block identifiers, it follows that action $\mathtt{move}(\mathsf{a},\mathsf{d},\mathsf{c})$ cannot be executed in the state depicted in Figure 2, in which block d represents the floor. We assume background knowledge such that, for example, $\mathtt{on}(\mathsf{b},\mathsf{c})$ implies $\neg\mathtt{clear}(\mathsf{c})$.

Suppose now the robotic arm to be in state of executing action $\mathtt{move}(\mathsf{a},\mathsf{d},\mathsf{c})$ also if block a is not clear, thus possibly moving a whole stack of blocks at one

[2] Therefore, when we talk about norm implementation we are not referring to the term implementation in its programming acception like, for instance, in [31].

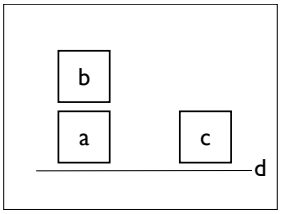

Fig. 2. Initial state.

time. Suppose also that the system designer considers such actions as undesirable. In this case the robotic arm can be considered as an autonomous agent, and the designer as a legislator or policymaker. In order to keep the example perspicuous, the scenario is limited to one agent, but we can express multiple agents analogously. The action $\texttt{move}(\mathsf{a},\mathsf{d},\mathsf{c})$ would get the following specification. Formula (4) does no longer demand $\texttt{clear}(\mathsf{a})$, but Formula (5) does not specify the effect when this is the case, i.e., when two or more blocks are moved simultaneously.

$$(\texttt{on}(\mathsf{a},\mathsf{d}) \wedge \texttt{clear}(\mathsf{c}) \wedge \texttt{turn}(i)) \leftrightarrow \langle \texttt{move}(\mathsf{a},\mathsf{d},\mathsf{c})(i)\rangle \top \tag{4}$$

$$\begin{aligned}(\texttt{on}(\mathsf{a},\mathsf{d}) \wedge \texttt{clear}(\mathsf{c}) \wedge \texttt{clear}(\mathsf{a}) \wedge \texttt{turn}(i)) \rightarrow\ & [\texttt{move}(\mathsf{a},\mathsf{d},\mathsf{c})(i)](\texttt{on}(\mathsf{a},\mathsf{c}) \\ & \wedge \texttt{clear}(\mathsf{a}) \\ & \wedge \neg \texttt{clear}(\mathsf{c})) \end{aligned} \tag{5}$$

$$\begin{aligned}(\texttt{on}(\mathsf{a},\mathsf{d}) \wedge \texttt{clear}(\mathsf{c}) \wedge \neg\texttt{clear}(\mathsf{a}) \wedge \texttt{turn}(i)) \rightarrow\ & [\texttt{move}(\mathsf{a},\mathsf{d},\mathsf{c})(i)](\texttt{on}(\mathsf{a},\mathsf{c}) \\ & \wedge \neg \texttt{clear}(\mathsf{a}) \\ & \wedge \neg \texttt{clear}(\mathsf{c}) \\ & \wedge \texttt{viol}(i)) \end{aligned} \tag{6}$$

where $\texttt{viol}(i)$ intuitively denotes a state brought about by agent i which is undesirable from the point of view of the system designer.

Suppose also that the system designer wants to implement the norm expressed by Formula (6).[3] The paper tackles this question displaying a number of strategies for norm implementation (Sections 4, 5, 6 and 7).

3.4 Talking about norms and extensive games in the Blocks World

In this section we bring together the logic-based perspective on norms sketched in Section 3.1 with the game-theoretic setting argued for in Section 3.2. This will be done in the context of the Blocks World scenario of the previous section. As a result we obtain a very simple modal logic language[4] which suffices to

[3] Notice that Formula (6) is an instance of Formula (1).

[4] For a comprehensive exposition of modal logic the reader is referred to [6].

express the properties of extensive games relevant for the purpose of the norm implementation analysis of the Blocks World.

Language. The language is the standard propositional modal logic language with n modal operators, where $n = |\texttt{Act}|$, that is, one modal operator for each available transition label. In addition, the non-logical alphabet of the language, consisting of the set of atomic propositions $\texttt{Pr}$ and of atomic actions $\texttt{Act}$, contains at least:

- Atoms in $\texttt{Pr}$ denoting game structure: for all agents $i \in I$, $\texttt{turn}(i)$, $\texttt{payoff}(i, x)$, labeling those states where it is player's i turn, and where the payoff for player i is x, where x is taken from a finite set of integers. The set of atoms denoting payoffs is referred to as $\texttt{Pr}^{\texttt{pay}}$.
- Atoms in $\texttt{Pr}$ denoting Blocks World states-of-affairs: for all blocks $\mathsf{a}, \mathsf{b} \in B$,$\texttt{on}(\mathsf{a}, \mathsf{b})$, $\texttt{clear}(\mathsf{a})$, labeling those states where block a is on block b, and where block a has no block on it.
- Atoms in $\texttt{Pr}$ denoting normative states-of-affairs: for all agents $i \in I$, $\texttt{viol}(i)$, labeling those states where player i has committed a violation.
- Atoms in $\texttt{Act}$ denoting deterministic transitions: for all agents $i \in I$ and blocks $\mathsf{a}, \mathsf{b}, \mathsf{c} \in B$: $\texttt{move}(\mathsf{a}, \mathsf{b}, \mathsf{c})(i)$, labeling those state transitions where player i moves block a from the top of block b to the top of block c.

The inductive definition of the set of formulae obtained from compounding via the set of Boolean connectives $\{\bot, \neg, \wedge\}$ and the modal connectives $\{\langle \mathbf{a} \rangle\}_{\mathbf{a} \in \texttt{Act}}$ is the standard one.

Semantics. Models are labelled transition systems $m = \langle W, W_{end}, \{R_a\}_{a \in \texttt{Act}}, \mathcal{I} \rangle$ such that:

- W is a non-empty set of system states;
- $\{R_a\}_{a \in \texttt{Act}}$ is a family of labelled transitions forming a rooted finite tree, i.e. there is a node such that there is a unique path from this node to all other nodes;
- W_{end} are the leaves of the finite tree;
- $\mathcal{I} : \texttt{Pr} \longrightarrow 2^W$ is the state labeling function.

The standard satisfaction relation $\models$ between pointed models (m, w) and modal formulae is assumed [6]. In addition, the models are assumed to satisfy the determinism condition, for all $\mathbf{a} \in \texttt{Act}$:

$$\langle \mathbf{a} \rangle \phi \rightarrow [\mathbf{a}] \phi.$$

Please note that such a condition is typical of the representation of actions within games in extensive form.[5]

Now everything is put into place to formulate with exactness the question that will be addressed in the next sections. Consider a model m as represented

[5] The reader is referred, for more details, to [4].

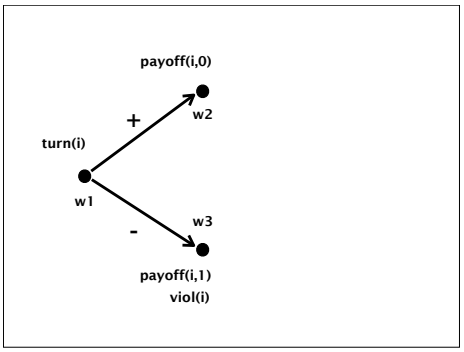

Fig. 3. Initial model.

in Figure 3. State w_1 is assumed to satisfy the relevant Blocks World description of Figure 2: $(m, w_1) \models \mathtt{on}(\mathsf{a}, \mathsf{d}) \wedge \mathtt{clear}(\mathsf{c}) \wedge \mathtt{clear}(\mathsf{b}) \wedge \neg\mathtt{clear}(\mathsf{a})$.[6] Notice that in the model it is also assumed that agent i leans towards executing the action "$-$" leading to the $\mathtt{viol}(i)$-state which has got a higher payoff. The actions and the violation conditions are given in the norm implementation problem. For example, there may have been an obligation to do $+$, a prohibition to do $-$, an obligation to reach state w_2, or a prohibition to reach w_3. Which norm is created is part of the norm synthesis or creation problem, but not part of the norm implementation problem. For the latter problem discussed in this paper, the structure of Figure 3.

Consider now a normative specification as represented by formulae like Formula (6), together with an initial model (such as the one in Figure 3). What are the transformations of the model m, guaranteeing that the agents in the system will comply with the normative specification? This is, in a nutshell, what we are going to investigate in the remainder of the paper.

3.5 Two important caveats

Before starting off with our analysis, we find it worth stating explicitly also what this work is not about.

The issue of norm implementation as intended here has already received attention in the literature on MAS in the form of the quest for formal languages able to specify sanctioning and rewarding mechanisms to be coupled with normative systems specifications. An example in this sense—but not the unique

[6] To avoid clutter in figures and notation, in what follows forbidden actions (e.g. $\mathtt{move}(\mathsf{a}, \mathsf{d}, \mathsf{c})(i)$ at w_1) are denoted by "$-$", and allowed actions (e.g. $\mathtt{move}(\mathsf{b}, \mathsf{a}, \mathsf{c})(i)$ at w_1) are denoted by "$+$". We are confident that this notational simplification will not give rise to misunderstandings.

one—is [50], where authors are concerned with the development of a whole framework for the formal specification of NMAS. Such a framework is able to capture also norm-implementation mechanisms such as sanctioning and rewarding systems. As our research question discussed in Section 1 shows, our aim in this paper differs from all such studies which can be found in the literature. The purpose of the paper is not to develop a formalism for the specification of one or another mechanism which could be effectively used for implementing norms in MAS. Rather, the paper aims at moving a first step towards the development of a comprehensive formal theory of norm implementation. Such a theory should be able to capture all forms of norm implementation mechanisms highlighting their common features and understanding them all as system transformations.

Finally, we want to stress that the present contribution abstracts completely from the issue concerning the motivating aspect of norms, that is to say, their capacity to influence and direct agents' mental states and actions. We are not assuming here that agents have the necessary cognitive capabilities to autonomously accept or reject norms [24]. To put it yet otherwise, the perspective assumed here is the one of a social designer aiming at regulating a society of agents by just assuming such agents to be game-theoretic agents. We are of course aware of this simplification, which is on the other hand necessary as we are facing the very first stage of the development of a formal theory.

4 Making violations impossible

The present sections studies two simple ways of making illegal states unreachable within the system.

4.1 Regimentation

Regimentation [46] is the simplest among the forms of implementation. Consider our running example, and suppose the social designer wants to avoid the execution of $\mathtt{move}(\mathsf{a},\mathsf{d},\mathsf{c})$ by (i) in the case block a is not clear, as expressed in Formula (6).

The implementation via regimentation for a transition $\mathbf{a}$ can be represented by a transformation (or update) $m \longmapsto m'$ of the model m into the model m' such that:

$$R^{m'}_{\mathbf{a}} := R^{m}_{\mathbf{a}} - \{(w,w') \mid (m,w) \models \mathtt{pre}_{\mathbf{a}} \;\&\; (m,w') \models \mathtt{viol}(i)\}$$

where $\mathtt{pre}_{\mathbf{a}}$ are the preconditions of the execution of $\mathbf{a}$ leading to a violation. In other words, it becomes in m' impossible to execute a transition with label $\mathbf{a}$ in $\mathtt{pre}_{\mathbf{a}}$-states leading to a violation state.

In the running example, where $\mathbf{a} = \mathtt{move}(\mathsf{a},\mathsf{d},\mathsf{c})(i)$, such update results in pruning away the edges labeled by "$-$" (i.e., $\mathtt{move}(\mathsf{a},\mathsf{d},\mathsf{c})(i)$ form the frame of m (Figure 4). The regimentation of the prohibition expressed in Formula (6) corresponds therefore to the validity of the following property:

$$(\mathtt{on}(a,d) \wedge \mathtt{clear}(\mathsf{c}) \wedge \neg\mathtt{clear}(\mathsf{a}) \wedge \mathtt{turn}(i)) \rightarrow [\mathtt{move}(a,d,c)(i)]\bot$$

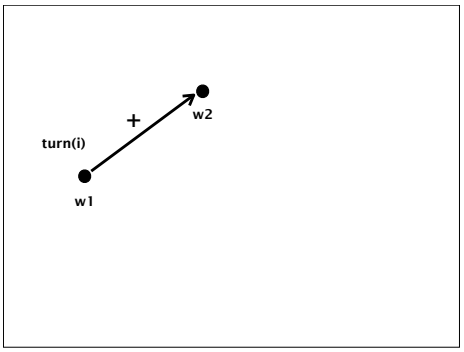

Fig. 4. Regimentation.

and hence, by modal logic and some additional background knowledge on the Blocks World:

$$(\mathtt{on}(a,d) \wedge \mathtt{clear(c)} \wedge \neg \mathtt{clear(a)} \wedge \mathtt{turn}(i)) \leftrightarrow \langle \mathtt{move}(a,d,c)(i) \rangle \top$$

which, notice, is a strengthening of Formula (4). In other words, regimentation is an update restricting the possibility of actions of the agents by limiting them exactly to the ones generating legal states. It is instructive to notice that the standard Blocks World scenario can be viewed precisely as a result of the regimentation of the normative variant of the scenario which we are considering here.

Within multi-agent systems, regimentation has been the first technique used for norm implementation. A typical example of this is AMELI [29], where all executable actions of agents are actions which are allowed according to the rule of the institutions. A formal semantics for a multi-agent program capturing regimentation is also studied in [26].

4.2 Retarded preconditions

Ordinary action logic describes the actions using preconditions and postconditions. If $\mathbf{a}$ is an action with precondition $\mathtt{pre}_{\mathbf{a}}$ and postcondition $\mathtt{post}_{\mathbf{a}}$ then

$$\mathtt{pre}_{\mathbf{a}} \leftrightarrow \langle \mathbf{a} \rangle \top \tag{7}$$

$$\mathtt{pre}_{\mathbf{a}} \rightarrow [\mathbf{a}]\mathtt{post}_{\mathbf{a}} \tag{8}$$

express that action $\mathbf{a}$ can be executed if and only if $\mathtt{pre}_{\mathbf{a}}$ hold (Formula (7)) and with the effect expressed by $\mathtt{post}_{\mathbf{a}}$ (Formula (8)). So in the standard account of the blocks world, if the world is in the situation as depicted in Figure 2, b can be moved on top of c but a cannot be moved.

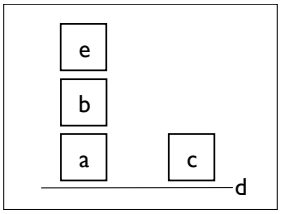

Fig. 5. Retarded preconditions. Initial state.

According to the normative perspective we have assumed in the running example, instead of imposing logically strong preconditions, we state logically weak preconditions for action, which means allow their execution in a wider range of states and assuming indeterminacy. In addition, we label states reached by performing actions as violation states when they are executed under undesirable conditions (see Section 3.3). In short, actions are allowed to be executed under circumstances which can possibly lead to violations, but only if the effects are still acceptable. If they are not, then nothing has happened.

These intuitions lead us to introduce, within the framework exposed in Section 3.4, two new modal operators: $\langle \phi \mid \mathbf{a} \rangle \ \psi$ and $[\phi \mid \mathbf{a}] \ \psi$. The semantics of these new operators is defined as follows:

$$m, w \models [\phi \mid \mathbf{a}] \ \psi \ \text{ iff } \ \forall w' \in W \text{ if } wR_{\mathbf{a}}|_{[\![\phi]\!]}w' \text{ then } m, w' \models \psi$$
$$m, w \models \langle \phi \mid \mathbf{a} \rangle \ \psi \ \text{ iff } \ \exists w' \in W \text{ such that } wR_{\mathbf{a}}|_{[\![\phi]\!]}w' \text{ and } m, w' \models \psi$$

where $[\![\phi]\!]$ denotes, as usual, the truth-set of ϕ and $R_{\mathbf{a}}|_{[\![\phi]\!]}$ is the subset of $R_{\mathbf{a}}$ containing those state pairs (w, w') such that the second element w' of the pair satisfies ϕ.[7] Notice, therefore, that the new modal operators take an action (e.g., $\mathbf{a}$) and a formula (e.g., ϕ) yielding a new complex action type (e.g., $\phi \mid \mathbf{a}$). Such action type corresponds, semantically, to those state transitions which are of the given action type ($\mathbf{a}$) and which end up in the given states (ϕ). So, retarded preconditions are represented, rather than as a formula, as part of an action type. This is in fact natural, as retarded preconditions are a way to further specify an action type.

Note that if we consider only a single update, then it would suffice to introduce $R_{\mathbf{a}}|_{[\![\phi]\!]}$ as the subset of $R_{\mathbf{a}}$ containing pairs (w, w') such that $w' \models \phi$. However, for sequential actions we have multiple updates and this would not suffice. This illustrates that we have a reduction from our logic to the fragment without the dynamic operators, as usually done in update logic (see, e.g., [28]).

[7] It might be instructive to notice that such operators are definable within standard dynamic logic [38] by means of the sequencing operator ; and the test operator ?: $[\phi \mid \mathbf{a}] \ \psi := [\mathbf{a}; ?\phi]\psi$. However, the full expressivity of dynamic logic is not required given our purposes.

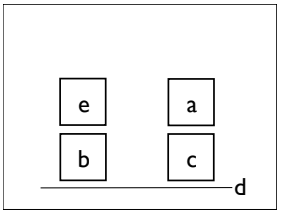

Fig. 6. Situation A

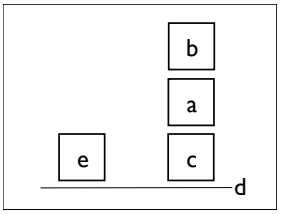

Fig. 7. Situation B

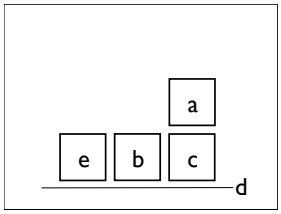

Fig. 8. Situation C

By means of this newly introduced operators, we can express that the execution of a given action **a** is possible only under the condition that it has certain precise effects ϕ (Formula (9)), and that each time it is executed having such effects ϕ, it also guarantees that ψ holds (Formula (10)):

$$\mathtt{pre}_{\mathbf{a}} \rightarrow \langle \mathtt{ret\_pre}_{\mathbf{a}} \mid \mathbf{a} \rangle \top \tag{9}$$

$$\mathtt{pre}_{\mathbf{a}} \rightarrow [\mathtt{ret\_pre}_{\mathbf{a}} \mid \mathbf{a}]\, \mathtt{post}_{\mathbf{a}} \tag{10}$$

where $\mathtt{pre}_{\mathbf{a}}$ represents the precondition of **a** where the execution of **a** possibly leads to a violation; $\mathtt{ret\_pre}_{\mathbf{a}}$ the postcondition of $\mathtt{pre}_{\mathbf{a}}$ which are tolerated, i.e., its *retarded preconditions*; and $\mathtt{post}_{\mathbf{a}}$ the postconditions of $\mathtt{ret\_pre}_{\mathbf{a}} \mid \mathbf{a}$.

Let us now give an example. Suppose we have the situation depicted in Figure 5. We move a, and we might end up with one of the three options in Figures 6-8. Suppose also that only the situation depicted in Figure 6 is tolerable to us. That is, a can be moved on c only if it is slid out carefully from the tower composed by a, b, e. Such tolerance can be expressed by means of *retarded precondition*, that is, a precondition which is evaluated as a result of the action performed. In the example at hand, the execution of action $\mathtt{move}(\mathsf{a}, \mathsf{d}, \mathsf{c})$ is tolerated in the case a is moved from within a tower only if the result of the action yields the situation depicted in Figure 6:[8]

$$\mathtt{on}(\mathsf{a}, \mathsf{d}) \wedge \mathtt{on}(\mathsf{e}, \mathsf{b}) \wedge \mathtt{clear}(\mathsf{c}) \rightarrow \langle \mathtt{on}(\mathsf{e}, \mathsf{b}) \mid \mathtt{move}(\mathsf{a}, \mathsf{d}, \mathsf{c}) \rangle \top \tag{11}$$

$$\mathtt{on}(\mathsf{a}, \mathsf{d}) \wedge \mathtt{on}(\mathsf{e}, \mathsf{b}) \wedge \mathtt{clear}(\mathsf{c}) \rightarrow [\neg \mathtt{on}(\mathsf{e}, \mathsf{b}) \mid \mathtt{move}(\mathsf{a}, \mathsf{d}, \mathsf{c})] \bot \tag{12}$$

$$\mathtt{on}(\mathsf{a}, \mathsf{d}) \wedge \mathtt{on}(\mathsf{e}, \mathsf{b}) \wedge \mathtt{clear}(\mathsf{c}) \rightarrow [\mathtt{on}(\mathsf{e}, \mathsf{b}) \mid \mathtt{move}(\mathsf{a}, \mathsf{d}, \mathsf{c})] \mathtt{on}(\mathsf{a}, \mathsf{c}). \tag{13}$$

Block a can be moved also in the case it is not clear, provided that this does not change the respective disposition of other blocks b and e (Formula 11). If that is not the case, than it will not be possible to move it (Formula 12). The effect of the execution of a under the retarded precondition that the stack of b and e is left intact results in a being placed on c (Formula 13).

The specification of retarded preconditions for actions can be viewed as a smoothening of regimentation requirements. As shown in the example above,

[8] We drop the $\mathtt{turn}(i)$ atoms in the following formalization.

instead of regimenting the non-execution of action $\mathtt{move}(\mathtt{a},\mathtt{d},\mathtt{c})$ in case block $\mathtt{a}$ as positioned within a tower, we can express that the execution can be tolerated, provided it gives rise to specific results (Figure 6).

In a nutshell, the use of retarded preconditions is typical of situations where the execution of a given action $\mathbf{a}$ under certain circumstances ϕ can possibly lead to a violation state:

$$\phi \wedge \langle \mathbf{a} \rangle \mathtt{viol}.$$

In such cases, we might not want to impose a regimentation, requiring that:

$$\phi \rightarrow [\mathbf{a}]\bot$$

but we would rather still allow the agent to perform the action, provided that it does not end up in violation states, that is, we allow the execution of the action under the potentially problematic conditions ϕ but only by assuming the retarded precondition $\neg\mathtt{viol}$:

$$\phi \rightarrow \langle \neg\mathtt{viol} \mid \mathbf{a} \rangle \top$$
$$\phi \rightarrow [\mathtt{viol} \mid \mathbf{a}]\bot$$

We conclude spending a few more words on the notion of retarded precondition. Such notion of retarded precondition is implicit in our culture. The saying "you can't argue with success" illustrates that way of thinking. An agent can take action without following the rules and if he is successful then we have to accept it. A major example is Admiral Nelson defying command and defeating the Spanish fleet. He is a hero. Had he failed, he would have been court marshalled.

5 Perfect enforcement

Perfect enforcement takes place when the execution of an action leading to a violation state is directly deterred by modifying the payoffs that the agent would obtain from such an execution. The following condition says that the best action does not imply a violation of the norm. It covers both penalties and rewards, or combinations of them.

Let $\mathtt{Pr}^{\mathtt{pay}}$ denote the set of payoff atoms and let $Max(i)$ denote the maximum payoff an agent i gets at a violation end state, if such a state exists. The implementation for i via perfect enforcement with respect to a transition $\mathbf{a}$, is a model update changing $m = (W, \{R_a\}_{a\in\mathtt{Act}}, \mathcal{I})$ to $m' = (W', \{R'_a\}_{a\in\mathtt{Act}}, \mathcal{I}')$ as follows:

- $W = W'$;
- $W_{end} = W'_{end}$;
- $\{R_a\}_{a\in\mathtt{Act}} = \{R'_a\}_{a\in\mathtt{Act}}$;
- $\mathcal{I}^m\lceil \mathtt{Pr} - \mathtt{Pr}^{\mathtt{pay}} = \mathcal{I}^{m'}\lceil \mathtt{Pr} - \mathtt{Pr}^{\mathtt{pay}}$, where $\lceil$ denotes the domain restriction operation on functions;
- $\mathcal{I}^{m'}\lceil \mathtt{Pr}^{\mathtt{pay}}$ is such that if $W_{end} \cap -\mathcal{I}(\mathtt{viol}(i)) \neq \emptyset$, then for some payoff atom $\mathtt{payoff}(i,x)$ with $x > Max(i)$ and state $w \in W'_{end}$, $w \in \mathcal{I}'(\mathtt{payoff}(i,x))$.

Notice that the update does not modify the interpretation of atoms which are not payoff atoms nor the frame of the model. What it does is to change $\mathcal{I}$ to a valuation $\mathcal{I}'$ which guarantees that at least one state in the end states of the game which are not violation states (if such states exist), the payoff for i is higher than the payoff in any of the violation states. Intuitively, such an update guarantees that each agent faced with a decision between executing a transition leading to a violation state, and one leading to a legal one, will—if they act rationally from a decision-theoretic perspective—choose for the latter.

A number of different implementation practices can be viewed as falling under this class such as, for instance, fines or side payments. However, the common feature consists in viewing the change in payoffs as infallibly determined by the enforcement, thereby giving rise to perfect deterrence. The next section will show what happens if such an assumption is dropped.

Getting back to our running example, the perfect enforcement of the prohibition expressed in Formula (6) results, therefore, in the validity of the following property:

$$\texttt{on}(\mathsf{a},\mathsf{d}) \wedge \texttt{clear}(c) \wedge \texttt{on}(\mathsf{b},\mathsf{a}) \wedge \texttt{turn}(i) \rightarrow ([+]\texttt{payoff}(i,1) \wedge [-]\texttt{payoff}(i,0))$$

where $+ = \texttt{move}(\mathsf{b},\mathsf{a},\mathsf{c})(i)$ and $- = \texttt{move}(\mathsf{a},\mathsf{d},\mathsf{c})(i)$.

We deem worth stressing again the subtle difference between perfect enforcement and regimentation. While regimentation makes it impossible for the agents to reach a violation state, automatic enforcement makes it just irrational in a decision-theoretic sense. In other words, it is still possible to violate the norm, but doing that would be the result of an irrational choice. As such, perfect enforcement is the most simple form of implementation which leaves the game form (i.e., the frame of the modal logic models) intact. Although the extensive game considered is a trivial one-player game, it should be clear that taking more player into consideration would not be a problem. In such case, the application of solution concepts such as sub-game perfect Nash [56] would become relevant.

In the multi-agent systems literature, perfect enforcement is used to provide a formal semantics to multi-agent programs in [26].

6 Enforcers

Commonly, perfect deterrence is hard to realize as each form of sanctioning requires the action of some third-party agent whose role consists precisely in making the sanctions happen. Enforcement via agents (the enforcers) corresponds to the update of model m to a model m' defining a new game form between a player i and enforcer j. The actions of enforcer j are $\texttt{punish}(i)$ and $\texttt{reward}(i)$. As a result of such an update, the original model m results in a sub-model of m'. The update is defined as follows:

- $W'_{end} = \{(w,n) \mid w \in W_{end} \;\&\; n \in \{1,2\}\}$;

- $W' = W \cup W'_{end}$, that is, each dead end of W is copied twice and added to the domain;
- $\{R'_a\}_{a \in \texttt{Act}} = \{R_a\}_{a \in \texttt{Act}}$, that is, the labeling via $\texttt{Act}$ remains the same;
- $R'_{\texttt{reward}(i)} = \{(w, w') \mid w \in W_{end} \;\&\; w' = (w, 1)\}$ and $R'_{\texttt{punish}(i)} = \{(w, w') \mid w \in W_{end} \;\&\; w' = (w, 2)\}$, that is, the added transitions are labeled as rewarding and punishing;
- $\mathcal{I}' = \mathcal{I}$ for all states in W;
- $\mathcal{I}'$ for the states in W'_{end} is such that $\forall w, w'$ s.t. $w \in W_{end}$ and $(m', w) \models \texttt{payoff}(i, x))$ and $wR^{m'}_{\texttt{reward}(i)}w' : (m', w') \models \texttt{payoff}(i, y)$ with $x \leq y$; and $\forall w, w'$ s.t. $w \in W_{end}$ and $(m', w) \models \texttt{payoff}(i, x))$ and $wR^{m'}_{\texttt{punish}(i)}w' : (m', w') \models \texttt{payoff}(i, y)$ with $x > y$;

What the definition above states is that the update consists in adding to every dead end in m a trivial game consisting of a binary choice by enforcer j between punishing or rewarding agent i. The result of a reward leaves the payoff of i intact (or it increases it), while the result of a punishment changes i's payoff to a payoff which is lower than the payoff i would have obtained by avoiding to end up in a violation state. In the running example, the action of the enforcer swaps the payoffs of agent i from 0 to 1 or from 1 to 1 in case of a reward; from 1 to 0 or from 0 to 0 in case of a punishment, just like in the case of automatic enforcement.

However, the use of agents as enforcers implies the introduction of a further normative level, since the enforcer can choose whether to comply or not with its role, that is, punish if i defects, and reward if i complies:

$$(\texttt{turn}(j) \wedge \texttt{viol}(i)) \rightarrow [\texttt{reward}(i)]\texttt{viol}(j) \tag{14}$$

$$(\texttt{turn}(j) \wedge \neg\texttt{viol}(i)) \rightarrow [\texttt{punish}(i)]\texttt{viol}(j) \tag{15}$$

Whether the enforcement works or not, depends on the payoffs of the enforcer j. We are, somehow, back to the original problem of guaranteeing the behavior of an agent (the enforcer in this case) to comply with the wishes of the social designer. The implementation of norms calls for more norms (Figure 9). Enforcement via enforcing agents lifts the implementation problem from the primary norms addressed to the agents in the system, to norms addressed to special agents with 'institutionalized' roles.

To the best of our knowledge, the only systematic multi-agent system frameworks addressing norm implementation at the level of enforcement is $\mathcal{M}$oise$^+$ and its variants (e.g., [42]), although not providing a formal semantics for it.

6.1 Regimenting enforcement norms.

At this point, the norms expressed in Formulae (14) and (15) need implementation. Again, regimentation can be chosen. The result of regimentation of enforcement norms in the running example is depicted in Figure 10. Formally, this

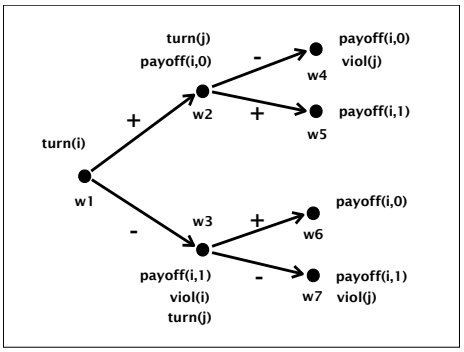

Fig. 9. Enforcement norms.

corresponds to an update $m \longmapsto m'$ of m where:

$$R^{m'}_{\texttt{punish(i)}} = R^{m}_{\texttt{punish(i)}} - \{(w, w') \mid m, w \models \texttt{turn}(j) \wedge \texttt{viol}(i)\ \&\ m, w' \models \texttt{viol}(j)\}$$
$$R^{m'}_{\texttt{reward(i)}} = R^{m}_{\texttt{reward(i)}} - \{(w, w') \mid m, w \models \texttt{turn}(j) \wedge \neg\texttt{viol}(i)\ \&\ m, w' \models \texttt{viol}(j)\}$$

As a result, the enforcer j always complies with what expected from its role. In a way, regimented enforcement can be viewed as an equivalent variant of perfect enforcement since its result is an adjustment of the payoffs of agent i w.r.t. to the system's norms.

6.2 Enforcing enforcement norms.

If the payoffs of the enforcer are appropriately set in order for the game to deliver the desired outcome, then the system is perfectly enforced by enforcer j who autonomously complies with the enforcement norms expressed in Formulae (14) and (15), punishing player i when i commits a violation and rewarding i when i complies (Figure 11). In the running example, perfect enforcement of enforcement norms can be defined by a simple update $m \longmapsto m'$ of the interpretation functions of the two models such that:

$$\mathcal{I}^{m'}(\texttt{payoff}(j, 0)) = \mathcal{I}^{m}(\texttt{viol}(j))$$
$$\mathcal{I}^{m'}(\texttt{payoff}(j, 1)) = W - \mathcal{I}^{m}(\texttt{viol}(j))$$

which results in a perfect match between higher payoffs and legal behavior. Figure 12 represents, in strategic form, the extensive game depicted in Figure 11 between player i and enforcer j. It is easy to see that the desired outcome in which both i and j comply is the only Nash equilibrium [56]. It goes without saying that much more complex game forms could be devised, and different equilibrium notions could be chosen for norm implementation purposes. It is

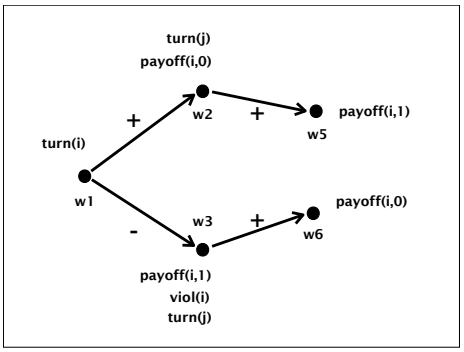

Fig. 10. Regimentation of enforcement norms.

at this level that a number of concepts and techniques could be imported from Mechanism Design and Implementation Theory [43–45, 52] to the formal theory of NMAS.

6.3 Who controls the enforcers?

Our analysis clearly shows the paradox hiding behind norm implementation. In order to implement norms, it is likely to need more norms.

The implementation of a set of norms can be obtained either via regimentation or via automatic enforcement or by the specification of an enforcement activity to be carried out by an enforcer. Enforcement specification happens at a normative level, i.e., via adding more norms to the prior set which, in turn, also require implementation. Schematically, suppose X to be the non-empty set of to-be-implemented norms, $Regiment(X)$ to denote the set of norms from X which are regimented or automatically enforced, and $Enforce(X)$ to denote the set of norms containing X together with all the norms specifying the enforcement of X ($X \subseteq Enforce(X)$). The implementation of S is the enforcement of the norms in S which are not regimented: $Implement(X) = Enforce(X \setminus Regiment(X))$.

In other words, to implement a set of norms amounts to implement the set of unregimented norms together with their enforcement. These observations clearly suggest that the implementation of a set of norms yields a set of norms. Somehow, it is very difficult to get rid of norms when trying to implement them. The only possibility is via full regimentation or automatic enforcement. If $Regiment(X) = X$ then there is no norm left to be implemented. Instead if $Regiment(X) \subset X$ then $\emptyset \subset Implement(S)$, which means that the implementation operation should be iterated on $Implement(X)$. In principle, such iteration is endless, unless there exists a final implementation level whose norms are all regimented or automatically enforced.

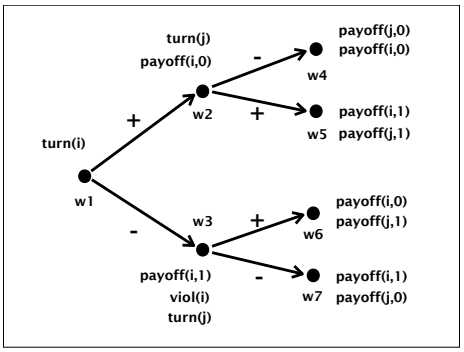

Fig. 11. Perfect enforcement.

i \ j	-	+
-	(1,0)	(0,1)
+	(0,0)	(1,1)

Fig. 12. Enforcement of the Blocks World scenario in strategic form

7 Implementation via norm change

This section concerns the ways of obtaining desired social outcomes by just modifying the set of norms of the system. The formal analysis of such phenomena, which is pervasive in human normative systems, is strictly related with the formal study of counts-as [34] and intermediate concepts [49].

As an example, consider the model m' obtained via the update of the initial model m corresponding to perfect enforcement (Figure 11). Suppose now the social designers wants to punish player i no matter what it does. One way for doing this would be to go back to the initial model m, to replace the enforcer norms expressed in Formulae (14) and (15) by the following norm:

$$\texttt{turn}(j) \rightarrow [\texttt{reward}(i)]\texttt{viol}(j) \tag{16}$$

and then update m to implement the norm expressed in Formula (16), for instance via perfect enforcement.

A much quicker procedure would consist in updating model m' trying to inherit its implementation mechanism. This can be done by simply modifying the extension of atom $\texttt{viol}(i)$ in order for it to include state w_2, thereby automatically triggering the enforcement norms expressed in Formulae (14) and (15). As a result, the enforcement mechanism in place in model m' are imported "for

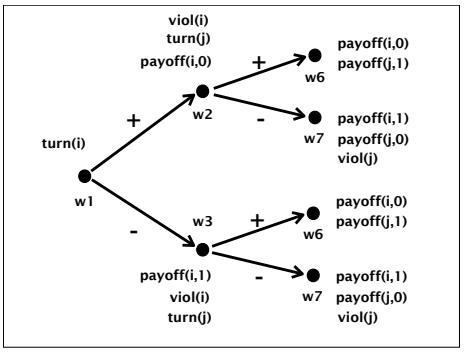

Fig. 13. Implementation via norm change.

free" by simply changing the meaning of $\texttt{viol}(i)$ (Figure 13). As you can see, the payoffs for enforcer j are different from Figure 11.

The update of the extension of $\texttt{viol}(i)$ can be obtained, for instance, by adding the following norm to the system:

$$\texttt{on}(\mathsf{b},\mathsf{a}) \wedge \texttt{clear}(\mathsf{b}) \wedge \texttt{clear}(\mathsf{c}) \wedge \texttt{turn}(i) \rightarrow [\texttt{move}(\mathsf{b},\mathsf{a},\mathsf{c})(i)]\texttt{viol}(i) \quad (17)$$

To put it otherwise, such procedure exploits the nature of $\texttt{viol}(i)$ as an intermediate concept occurring as precondition of other norms. In this case the norms involved are the enforcement norms expressed in Formulae (14) and (15).

8 Related work

In this section we consider whether existing work in normative multi-agent systems is able to answer the equation discussed in the introduction.

$$\text{BDI} : \text{Agent Programming} = ? : \text{NMAS Programming}.$$

Since BDI-CTL [23] is used as a formal specification and verification language for agent programming, an obvious candiate for our question mark is an extension of this language with deontic concepts such as obligations and permissions, called BOID-CTL [20,21]. Such a logic is simply a modal combination of an agent logic and a modal deontic logic. The drawback of this approach is that the norms are nor represented explicitly.

The first candidate for the question mark is Tennenholtz and Shoham's game-theoretic approach to artificial social systems. However, the central research question of their work [62,63,65] consists in studying the emergence of desirable social properties under the assumption that a given social law is followed by

the agents in the society at hands. The problem of how a social law can be implemented in the society is not discussed.

Another obvious candidate for the question mark is a theory of normative systems [1]. The key feature of normative systems is that they make norms explicit in such a way that we can say, at a given state of the system, whether a norm is active, in force, violated, and so on [67]. See [36] for an up to date review on the distinction between a theory of normative systems and deontic logic, and the challenge to bridge the two. A theory of normative systems is useful for norm representation and reasoning, but not for the representation of aspects such as the multi-agent structure of a normative system.

A third candidate is Boella and van der Torre's game-theoretic approach to normative multi-agent systems, which studies the more general problem of norm creation [9, 11, 14]. For example, the introduction of a new norm with sanctions is modeled as enforceable norms in artificial social systems as the choice among various strategic games [10]. They focus in particular on the enforcement of norms using enforcers, and discuss the role of procedural norms to motivate the enforcers [13]. They consider the creation of a new norm into a system of norms, whereas in this paper we do not consider the effect of norm implementation on existing norms. They argue that the infinite regression of enforcers can be broken if we assume that enforcers control each other and do not cooperate [10]. Since they use strategic rather than extensive games they cannot distinguish some subtle features of implementation such as retarded preconditions. Moreover, they do not give a procedure to go from a norm to its implemented system. Finally, they do not consider other methods than sanctioning and rewarding to implement their norms. They do consider also cognitive extensions of their model, which we do not consider in this paper. See [14] for a detailed discussion on their approach.

There are many organizational and institutional theories, such as the ones proposed in [34], and there is a lot of work on coordination and the environment [25, 58]. Institutions are built using constitutive norms defining intermediate concepts. However, this work is orthogonal to the work presented in this paper in as much as, although sporadically addressing one or another form of implementation, it never aims at laying the ground of an overarching formal framework.

9 Conclusions

Aim of the paper is to illustrate how the issue of norm implementation can be understood in terms of transformations (updates) performed on games in extensive forms. The paper has sketched some of such updates by means of a toy example, the blocks world, and mapped them to norm implementation strategies, such as regimentation, automatic enforcement, enforcement via enforcers, and implementation via norm change. The full logical analysis (e.g., in a dynamic logic setting) of the update operations sketched here is future work. Such an analysis will make some intricacies of implementation explicit, such as,

for instance the fact that by implementing new norms, the implementation of other norms might end up being disrupted.

Moreover, we introduce two views on representing forbidden actions, the classical one in which the precondition has to be satisfied before the action can be executed, and one based on so-called retarded preconditions. The two views coincide if the language allows for action names, and we can include as part of the state a list of which actions are allowed in this state. This can be formalised by the predicate *allowed*(X), where X are names for actions. The *allowed*(X) predicate can be part of the preconditions of X. We can use the feedback arrows of retarded preconditions in Kripke models to change accessibility. This will implement the severed connections in the diagrams, and the semantics would then be *reactive Kripke models.* Consider for example the restriction "you should not take any action three times in a row." With retarded preconditions, we can do a "roll-back" when the action occurs three times in a row, whereas with regimentation we have to predict whether the action is going to be executed three times rather than two or four times. A further comparison of the two views is topic for further research.

Finally, topics for further research are also the development of a more detailed classification of norm implementation methods, the application of retarded preconditions to the analysis of ambiguous norms.

Acknowledgments. The authors would like to thank the reviewers of the volume for their helpful comments. Davide Grossi wishes to acknowledge support by *Ministère de la Culture, de L'Enseignement Supérieur et de la Recherche, Grand-Duché de Luxembourg* (grant BFR07/123) and by *Nederlandse Organisatie voor Wetenschappelijk Onderzoek* (VENI grant 639.021.816).

References

1. C. E. Alchourrón and E. Bulygin. *Normative Systems*. Springer Verlag, 1971.
2. A.R. Anderson. A reduction of deontic logic to alethic modal logic. *Mind*, 22:100–103, 1958.
3. J. van Benthem. Extensive games as process models. *Journal of Logic, Language and Information*, 11:289–313, 2002.
4. J. van Benthem. Logic in games. Lecture Notes of the ILLC graduate course on Logic, Language and Information, Universiteit van Amsterdam, Amsterdam, The Netherlands, 2005.
5. J. van Benthem, J. van Eijck, and V. Stebletsova. Modal logic, transition systems and processes. *Journal of Logic and Computation*, 4(5):811–855, 1994.
6. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, Cambridge, 2001.
7. G. Boella, G. Pigozzi, and L. van der Torre. Five guidelines for normative multi-agent systems. In *Proceedings of JURIX'09*, 2009.
8. G. Boella, G. Pigozzi, and L. van der Torre. Norms in computer science: Ten guidelines for normative multi-agent systems. In G. Boella, P. Noriega, G. Pigozzi, H. Verhagen,

editors, *Normative Multi-agent Systems*, number 09121 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2009.
9. G. Boella and L. van der Torre. Δ: The social delegation cycle. In *Deontic Logic: 7th International Workshop on Deontic Logic in Computer Science (ΔEON'04)*, volume 3065 of *LNCS*, pages 29–42, Berlin, 2004. Springer.
10. G. Boella and L. van der Torre. Enforceable social laws. In *Procs. of 4th International Joint Conference on Autonomous Agents and multi-agent Systems (AAMAS'05)*, pages 682–689, New York (NJ), 2005. ACM Press.
11. G. Boella and L. van der Torre. Norm negotiation in multiagent systems. *Int. J. Cooperative Inf. Syst.*, 16(1):97–122, 2007.
12. G. Boella and L. van der Torre. The ontological properties of social roles in multi-agent systems: Definitional dependence, powers and roles playing roles. *Artificial Intelligence and Law Journal (AILaw)* 15(3): 201-221, 2007.
13. G. Boella and L. van der Torre. Substantive and procedural norms in normative multi-agent systems. *Journal of Applied Logic* 6(2): 152-171, 2008.
14. G. Boella and L. van der Torre. A game-theoretic approach to normative multi-agent systems. In G. Boella, L. van der Torre, and H. Verhagen, editors, *Normative Multi-agent Systems*, number 07122 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.
15. G. Boella, L. van der Torre, and H. Verhagen. Introduction to normative multi-agent systems. *Computational and Mathematical Organization Theory*, 12(2-3):71–79, 2006.
16. G. Boella, L. van der Torre, and H. Verhagen, editors. *Normative Multi-Agent Systems*, number 07122 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.
17. G. Boella, H. Verhagen, and L. van der Torre. Introduction to the special issue on normative multi-agent systems. *Journal of Autonomous Agents and Multi Agent Systems*, 17(1):1–10, 2008.
18. G. Boella, L. van der Torre, and S. Villata. Conditional dependence networks in requirements engineering. In *Proceedings of COIN'09*, LNCS. Springer, 2009.
19. W. Briggs and D. Cook. Flexible social laws. In *Proceedings 14th International Joint Conference on Artificial Intelligence*, pages 688–693, 1995.
20. J. Broersen, M. Dastani, J. Hulstijn, and L. van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3-4):428–447, 2002.
21. J. Broersen, M. Dastani, and L. van der Torre. Bdioctl: Obligations and the specification of agent behavior. In *Proceedings of IJCAI'03*, pages 1389–1390, 2003.
22. E. Bulygin. Permissive norms and normative systems. In A. Martino and F. Socci Natali, editors, *Automated Analysis of Legal Texts*, pages 211–218. Publishing Company, Amsterdam, 1986.
23. P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.
24. R. Conte, C. Castelfranchi, and F. Dignum. Autonomous norm acceptance. In J. Müller, M. P. Singh, and A. S. Rao, editors, *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555, pages 99–112. Springer-Verlag: Heidelberg, Germany, 1999.
25. M. Dastani, F. Arbab, and F. S. de Boer. Coordination and composition in multi-agent systems. In *Procs. of AAMAS*, pages 439–446, 2005.
26. M. Dastani, D. Grossi, N. Tinnemeier, and J.-J. Meyer. Normative multi-agent programs and their logics. In Pigozzi G. Noriega P. Boella, G. and H. Verhagen, editors, *Normative Multi-agent Systems, Dagstuhl Seminar Proceedings*, volume 09121, 2008.

27. V. Dignum. *A Model for Organizational Interaction*. SIKS Dissertation Series, 2003.
28. H.P. van Ditmarsch, W. van der Hoek, and B.P. Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library*. Springer, 2007.
29. 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.
30. D. Fitoussi and M. Tennenholtz. Choosing social laws for multi-agent systems: Minimality and simplicity. *Artificial Intel ligence*, 119:61–101, 2000.
31. A. Garcia-Camino, P. Noriega, and J. A. Rodriguez-Aguilar. Implementing norms in electronic institutions. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multi-agent systems*, pages 667–673. ACM Press, 2005.
32. A. Giddens. *The Constitution of Society*. University of California Press, 1984.
33. E. Goffman. *The Presentation of Self in Everyday Life*. Doubleday, 1959.
34. D. Grossi. *Designing Invisible Handcuffs. Formal Investigations in Institutions and Organizations for Multi-agent Systems*. PhD thesis, Utrecht University, SIKS, 2007.
35. D. Grossi, H. Aldewereld, and F. Dignum. Ubi lex ibi poena. designing norm enforcement in electronic institutions. In V. Dignum, N. Fornara, P. Noriega, G. Boella, O. Boissier, E. Matson, and J. Vázquez-Salceda, J.zquez-Salceda, editors, *Proceedings of COIN@AAMAS'06*, volume 4386 of *LNCS*, pages 101–114. Springer, 2006.
36. J. Hansen. *Imperatives and Deontic Logic: On the Semantic Foundations of Deontic Logic*. University of Leipzig, 2008.
37. J. Hansen, G. Pigozzi, and L. van der Torre. Ten philosophical problems in deontic logic. In G. Boella, L. van der Torre, and H. Verhagen, editors, *Normative Multi-agent Systems*, volume 07122 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.
38. D. D. Harel, A.M.D. Kozen and J. Tiuryn. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic: Volume II: Extensions of Classical Logic*, pages 497–604. Reidel, Dordrecht, The Netherlands, 1984.
39. W. van der Hoek, M. Roberts, and M. Wooldridge. Social laws in alternating time: Effectiveness, feasibility, and synthesis. *Synthese*, 156(1), 2007.
40. M. Hollis. *Trust within reason*. Cambridge University Press, Cambridge, 1998.
41. J. F. Hübner, J. S. Sichman, and O. Boissier. Moise+: Towards a structural functional and deontic model for mas organization. In *Proceedings of the First Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, Bologna, Italy, July 2002. ACM Press.
42. J. F. Hubner, J. S. Sichman, and O. Boissier. Developing organised multiagent systems using the moise+ model: programming issues at the system and agent levels. *Int. J. Agent-Oriented Softw. Eng.*, 1(3/4):370–395, 2007.
43. L. Hurwicz. Optimality and informational efficiency in resource allocation processes. In K. Arrow, S. Karlin, and P. Suppes, editors, *Mathematical Methods in the Social Sciences*. Stanford University Press, 1960.
44. M. O. Jackson. A crash course in implementation theory. *Social Choice and Welfare*, 18:655–708, 2001.
45. M. O. Jackson. Mechanism theory. In U. Derigs, editor, *Encyclopedia of Life Support Systems*. EOLSS Publishers, 2003.
46. A. J. I. Jones and M. Sergot. On the characterization of law and computer systems. *Deontic Logic in Computer Science*, pages 275–307, 1993.
47. S. Kanger. New fondations for ethical theory. In R. Hilpinen, editor, *Deontic Logic: Introductory and Systematic Readings*, pages 36–58. Reidel Publishing Company, 1971.

48. D. Lewis. A problem about permission. In E. Saarinen, editor, *Essays in Honour of Jaakko Hintikka*, pages 163–175. D. Reidel, Dordrecht, 1979.
49. L. Lindahl and J. Odelstad. Open and closed intermediaries in normative systems. In T.M. van Engers, editor, *Proceedings of the Nineteenth JURIX Conference on Legal Knowledge and Information Systems (JURIX 2006)*, pages 91–100, 2006.
50. F. Lopez, M. Luck, and M. d'Inverno. A normative framework for agent-based systems. *Computational and Mathematical Organization Theory*, 12:227–250, 2006.
51. M. Luck, P. McBurney, and C. Preist. *Agent Technology: Enabling Next Generation Computing (A Roadmap for Agent Based Computing)*. AgentLink, 2003.
52. E. Maskin. Nash equilibrium and welfare optimality. *Review of Economic Studies*, 66:23–38, 1999.
53. J.-J.Ch Meyer. A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 29(1):109–136, 1988.
54. D. C. North. *Institutions, Institutional Change and Economic Performance*. Cambridge University Press, Cambridge, 1990.
55. S. Onn and M. Tennenholtz. Determination of social laws for multi-agent mobilization. *Artificial Intel ligence*, 95:155–167, 1997.
56. M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.
57. Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a BDI-architecture. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991.
58. A. Ricci, A. Omicini, and E. Denti. Activity theory as a framework for mas coordination. In *Procs. of ESAW'02*, pages 96–110, 2002.
59. D.W.P. Ruiter. A basic classification of legal institutions. *Ratio Juris*, 10(4):357–371, 1997.
60. S. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach*. Prentice Hall International, 2001.
61. Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 276–281, 1992.
62. Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73(1-2):231–252, 1995.
63. Y. Shoham and M. Tennenholtz. On the emergence of social conventions: Modeling, analysis and simulations. *Artificial Intelligence*, 94(1–2):139–166, 1997.
64. A. Strauss. *Negotiations: Varieties, Contexts, Processes and Social Order*. San Francisco, Jossey-Bass, 1978.
65. M. Tennenholtz. On stable social laws and qualitative equilibria. *Artificial Intelligence*, 102(1):1–20, 1998.
66. M. Tennenholtz. On social constraints for rational agents. *Computational Intelligence*, 15(4), 1999.
67. L. van der Torre and Y. Tan. Diagnosis and decision making in normative reasoning. *Artificial Intelligence and Law*, 7(1):51–67, 1999.