Ruling the Blocks World: Towards a Game Change Framework for Norm Implementation *

Davide Grossi[1], Dov Gabbay[2], 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

leon.vandertorre@uni.lu

Abstract

The norm implementation problem concerns how a 'social designer' can see to it that the agents comply with a given set of norms in a given multiagent system. In this paper we discuss how various ways to implement norms in a multiagent system can be distinguished in a formal game-theoretic framework. In particular, we show how different types of norm implementation can all be uniformly understood 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.

1 Introduction

Normative multi-agent systems (NMAS) [10] 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. In other words, designing a NMAS does

*normimpl.tex, *Tuesday 30th June, 2009, 18:54.*

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.

The paper moves the first steps towards a formal understanding of the norm implementation problem, defined as follows.

> *The norm implementation problem.* How can a given set of norms be made complied with by the agents in a given system?

The norm implementation problem is part of the more general problem 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 problem [43], 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. In this paper we introduce a formal framework that can represent these various solutions to the norm implementation problem, which can be used to analyze them, or to make a choice among them.

The assumption underlying our research problem is that the norm implementation problem can be studied in isolation. We thus argue against 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 between 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.

The reader may be surprised that the norm implementation problem has not been dealt with thus far, given the high number of publications on social, organizational and deontic theories recently. Ideas we discuss can be found scattered over this literature, but they have not been presented systematically and in a uniform framework. Our aim in this paper is to unify these ideas and present them in a more abstract way.

The first requirement on a framework for norm implementation is that it can represent existing widely discussed norm implementation methods such as regimentation and 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. 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. Which formal model to study the norm implementation problem?
2. How to model regimentation?
3. How to model enforcing?
4. How to model norm change?

The perspective assumed on the problem 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, or the composition of actions in plans. We consider only perfect information games, and we thus do not consider the problem how norm are distributed and communicated to a society. Moreover, we do not consider concurrent actions of the agents, although this feature could be easily added. However, we do represent the order of actions, that is, we use extensive form, 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 [43]). 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.

We test our model by introducing 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, assume that a block may not be puton another block if it stays there for three minutes. If it stayed there for three minutes, then we can undo the action (alternatively, one can sanction it, of course). 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.

It is our conviction that such approach will 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. [39]) have played for the programming

of single agents. Such an issue was recognized as central for the NMAS community during the NorMAS'07 Dagstuhl Seminar [13], and it was raised in the following incisive form:

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

This equation represents two issues. First, it raises the question about which concepts should be used for programming normative multiagent systems, 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 multiagent 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 multiagent structure of normative systems.

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. Alternative well known examples we could have picked is Wumpus from Russel and Norvig textbook [41].

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., [11]).

The paper follows the research questions and proceeds as follows. In Section 2 we start with the game-theoretic framework for norm implementation and a logic for representing extensive games, and we introduce a running example. Sections 3, 4, 6 provide formal semantics to the three implementation strategies of regimentation, enforcement and, respectively, normative change. The finding of each section will be illustrated by means of the running example. In Section 7 related work at the intersection of norms and multiagent systems is discussed. Conclusions follow in Section 8.

2 Formal framework and running example

The present section is devoted to setting the stage of our investigations.

2.1 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, 31, 35] 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:

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

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* [30] 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 [24].

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 [38]. The next section moves to the fundamental role that —we think— game theory can play for the analysis of the norm implementation problem.

2.2 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, as we will see, by means of other agents' actions. The readily available formal

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

framework to investigate this type of social interaction is, needless to say, game theory. In fact , 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 [27, 29, 34, 28], is concerned with the design of the interaction rules—the "rules of the game" [36] 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 [38]. 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 2.1, 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.

2.3 Running example: ruling the Blocks World

We assume a multiagent 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 [41], 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:

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

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

$$\wedge \mathtt{clear}(\mathsf{b}) \wedge \mathtt{clear}(\mathsf{a})) \tag{4}$$

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 1, in which block b represents the floor.

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 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 $\mathtt{move}(\mathsf{a},\mathsf{d},\mathsf{c})$ would get the following specification. Formula (5) does no longer demand $\mathtt{clear}(\mathsf{a})$, but Formula (5)

[2]Therefore, when we talk about norm implementation we are not referring to the term implementation in its programming acception like, for instance, in [22].

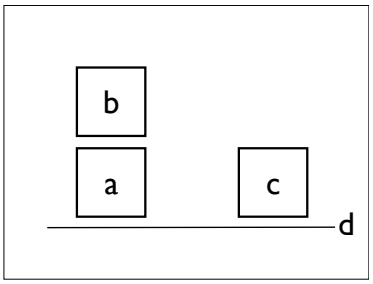

Figure 1: Initial state.

does not specify the effect when this is the case, i.e., when two or more blocks are moved simultaneously.

$$(\mathsf{on}(\mathsf{a},\mathsf{d}) \wedge \mathtt{clear}(\mathsf{c}) \wedge \mathtt{turn}(i)) \leftrightarrow \langle \mathtt{move}(\mathsf{a},\mathsf{d},\mathsf{c})(i)\rangle \top \quad (5)$$

$$(\mathsf{on}(\mathsf{a},\mathsf{d}) \wedge \mathtt{clear}(\mathsf{c}) \wedge \mathtt{clear}(\mathsf{a}) \wedge \mathtt{turn}(i)) \rightarrow [\mathtt{move}(\mathsf{a},\mathsf{d},\mathsf{c})(i)](\mathsf{on}(\mathsf{a},\mathsf{c}) \wedge \mathtt{clear}(\mathsf{a}) \wedge \neg\mathtt{clear}(\mathsf{c})) \quad (6)$$

$$(\mathsf{on}(\mathsf{a},\mathsf{d}) \wedge \mathtt{clear}(\mathsf{c}) \wedge \neg\mathtt{clear}(\mathsf{a}) \wedge \mathtt{turn}(i)) \rightarrow [\mathtt{move}(\mathsf{a},\mathsf{d},\mathsf{c})(i)](\mathsf{on}(\mathsf{a},\mathsf{c}) \wedge \neg\mathtt{clear}(\mathsf{a}) \wedge \neg\mathtt{clear}(\mathsf{c}) \wedge \mathtt{viol}(i)) \quad (7)$$

where $\mathtt{viol}(i)$ 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 (7)?[3] The paper tackles this question displaying a number of strategies for norm implementation (Sections 3, 4 and 6).

2.4 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 2.1 with the game-theoretic setting argued for in Section 2.2. 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 express the properties of extensive games relevant for the purpose of the norm implementation analysis of the Blocks World.

2.4.1 Language.

The language is the standard propositional modal logic language with n modal operators, where $n = |\mathtt{Act}|$, 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 $\mathtt{Pr}$ and of atomic actions $\mathtt{Act}$, contains at least:

[3] Notice that Formula (7) is an instance of Formula (1).

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

- Atoms in $\mathtt{Pr}$ denoting game structure: for all agents $i \in I$, $\mathtt{turn}(i)$, $\mathtt{payoff}(i, x)$, 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.
- Atoms in $\mathtt{Pr}$ denoting Blocks World states-of-affairs: for all blocks $\mathsf{a}, \mathsf{b} \in B$,$\mathtt{on}(\mathsf{a}, \mathsf{b})$, $\mathtt{clear}(\mathsf{a})$, labeling those states where block a is on block b, and where block a has no block on it.
- Atoms in $\mathtt{Pr}$ denoting normative states-of-affairs: for all agents $i \in I$, $\mathtt{viol}(i)$, labeling those states where player i has committed a violation.
- Atoms in $\mathtt{Act}$ denoting deterministic transitions: for all agents $i \in I$ and blocks $\mathsf{a}, \mathsf{b}, \mathsf{c} \in B$: $\mathtt{move}(\mathsf{a}, \mathsf{b}, \mathsf{c})(i)$, 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 $\{\langle \mathbf{a} \rangle\}_{\mathbf{a} \in \mathtt{Act}}$ is the standard one.

2.4.2 Semantics.

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

- W is a non-empty set of system states;
- $\{R_a\}_{a \in \mathtt{Act}}$ is a family of labelled transitions forming a tree;
- $\mathcal{I} : \mathtt{Pr} \longrightarrow 2^W$ 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 $\mathbf{a} \in \mathtt{Act}$:

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

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 in Figure 2. State w_1 is assumed to satisfy the relevant Blocks World description of Figure 1: $(m, w_1) \models \mathtt{on}(\mathsf{a}, \mathsf{d}) \wedge \mathtt{clear}(\mathsf{c}) \wedge \mathtt{clear}(\mathsf{b}) \wedge \neg \mathtt{clear}(\mathsf{a})$.[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.

Consider now a normative specification as represented by formulae like Formula (7), together with an initial model (such as the one in Figure 2). What are the transformations of the model m, which guarantee the normative specification to be complied with by the agents in the system? This is, in a nutshell, what we are going to investigate in the remainder of the paper.

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

[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.

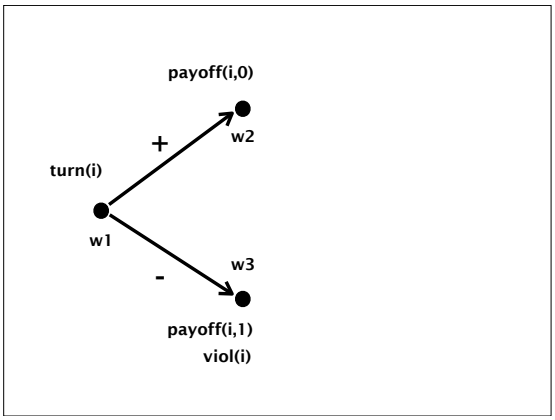

Figure 2: Initial model.

2.5 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 one—is [33], 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 [18]. 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.

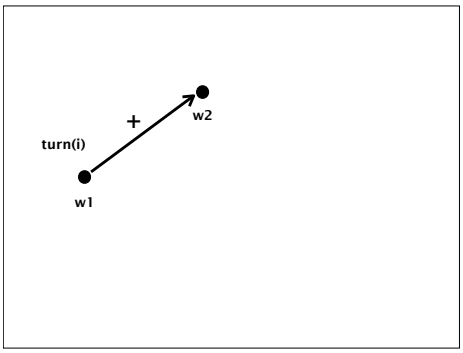

Figure 3: Regimentation.

3 Making violations impossible

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

3.1 Regimentation

Regimentation [30] is the most simple 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 (7).

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 $\mathbf{a}$ which lead 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 which lead to a violation state. Notice that if $\mathtt{pre}(\mathbf{a}) = \top$ then the update results in $R^{m'}_{\mathbf{a}} = \emptyset$.

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 3). The regimentation of the prohibition expressed in Formula (7) corresponds therefore to the validity of the following property:

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

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

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

which, notice, is a strengthening of Formula (5). In other words, regimentation is an update restricting the possibility of actions of the agents by limiting

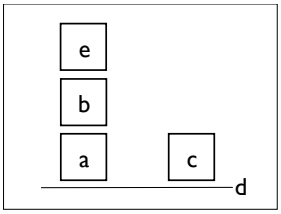

Figure 4: Retarded preconditions. Initial state.

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.

3.2 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

$$\mathtt{pre}_{\mathbf{a}} \quad \leftrightarrow \quad \langle \mathbf{a} \rangle \top \tag{8}$$

$$\mathtt{pre}_{\mathbf{a}} \quad \rightarrow \quad [\mathbf{a}]\mathtt{post}_{\mathbf{a}} \tag{9}$$

express that action $\mathbf{a}$ can be executed if and only if $\mathtt{pre}_{\mathbf{a}}$ hold (Formula (8)) and with the effect expressed by $\mathtt{post}_{\mathbf{a}}$ (Formula (9)). So in the standard account of the blocks world, if the world is in the situation as depicted in Figure 1, $\mathtt{b}$ can be moved on top of $\mathtt{c}$ but $\mathtt{a}$ cannot be moved.

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 2.3). 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 2.4, two new modal operators: $\langle \varphi \mid \mathbf{a} \rangle \, \psi$ and $[\varphi \mid \mathbf{a}] \, \psi$. The semantics of these new operators is defined as follows:

$$m, w \models [\varphi \mid \mathbf{a}] \, \psi \quad \text{iff} \quad \forall w' \in W \text{ if } wR_{\mathbf{a}}|_{[\![\varphi]\!]}w' \text{ then } m, w' \models \psi$$

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

where $[\![\varphi]\!]$ denotes, as usual, the truth-set of φ and $R_{\mathbf{a}}|_{[\![\varphi]\!]}$ 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.,

[7] It might be instructive to notice that such operators are definable within standard dynamic logic [19] by means of the sequencing operator ; and the test operator ?: $[\varphi \mid \mathbf{a}] \, \psi := [\mathbf{a}; ?\varphi]$. However, the full expressivity of dynamic logic is not required given our purposes.

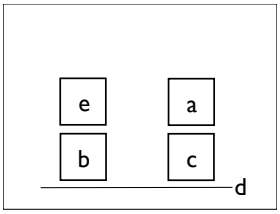

Figure 5: Situation A

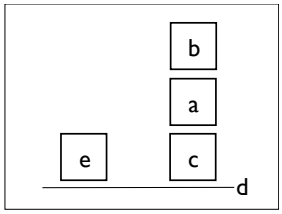

Figure 6: Situation B

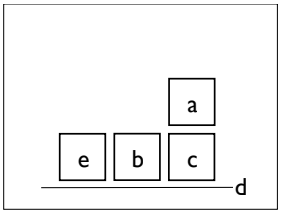

Figure 7: Situation C

a) and a formula (e.g., φ) yielding a new complex action type (e.g., $\varphi \mid \mathbf{a}$). Such action type correspond, semantically, to those state transitions which are of the given action type (**a**) and which end up in the given states (φ).

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 (10)), and that each time it is executed having such effects φ, it also guarantees that ψ holds (Formula (11)):

$$\mathtt{pre}_{\mathbf{a}} \rightarrow \langle \mathtt{ret\_pre}_{\mathbf{a}} \mid \mathbf{a} \rangle \top \tag{10}$$

$$\mathtt{pre}_{\mathbf{a}} \rightarrow [\mathtt{ret\_pre}_{\mathbf{a}} \mid \mathbf{a}]\, \mathtt{post}_{\mathbf{a}} \tag{11}$$

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 4. We move a, and we might end up with one of the three options in Figures 5-7. Suppose also that only the situation depicted in Figure 5 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 5:[8]

$$\mathsf{on}(\mathsf{a},\mathsf{d}) \wedge \mathsf{on}(\mathsf{e},\mathsf{b}) \wedge \mathtt{clear}(\mathsf{c}) \rightarrow \langle \mathsf{on}(\mathsf{e},\mathsf{b}) \mid \mathtt{move}(\mathsf{a},\mathsf{d},\mathsf{c}) \rangle \top \tag{12}$$

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

$$\mathsf{on}(\mathsf{a},\mathsf{d}) \wedge \mathsf{on}(\mathsf{e},\mathsf{b}) \wedge \mathtt{clear}(\mathsf{c}) \rightarrow [\mathsf{on}(\mathsf{e},\mathsf{b}) \mid \mathtt{move}(\mathsf{a},\mathsf{d},\mathsf{c})] \mathsf{on}(\mathsf{a},\mathsf{c}). \tag{14}$$

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 12). If that is not the case, than it will not be possible to move it (Formula 13). 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 14).

The specification of retarded preconditions for actions can be viewed as a smoothening of regimentation requirements. As shown in the example above, instead of regimenting the non-execution of action $\mathtt{move}(\mathsf{a},\mathsf{d},\mathsf{c})$ in case block a

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

as positioned within a tower, we can express that the execution can be tolerated, provided it gives rise to specific results (Figure 5).

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

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

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

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

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 $\neg\mathtt{viol}$:

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

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.

4 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 $R^m_{\mathtt{Act}} = \bigcup_{\mathbf{a}\in\mathtt{Act}} R_{\mathbf{a}}$, that is, $R^m_{\mathtt{Act}}$ labels the whole transition tree of model m. The implementation via perfect enforcement w.r.t. a transition **a** is a model update $m \longmapsto m'$ which does not modify the frame $(W, \{R_a\}_{a\in\mathtt{Act}})$ of the model, and only modifies the evaluation of payoff atoms[9] so that $\forall w \in W$:

If $\quad wR^{m'}_{\mathbf{a}}w'$ & $(m', w') \models \mathtt{viol}(i)) \wedge \mathtt{payoff}(i, x)$

then $\quad \exists w'' \in W$ such that $wR^{m'}_{\mathtt{Act}}w''$ & $(m', w'') \models \neg\mathtt{viol}(i)) \wedge \mathtt{payoff}(i, y)$

with $x < y$. Recall that the update does not modify the interpretation of atoms which are not payoff atoms nor the frames of the model (so, $R^{m'}_{\mathbf{a}} = R^m_{\mathbf{a}}$ for all $\mathbf{a} \in \mathtt{Act}$). Intuitively, such an update guarantees that each agent faced with a decision between executing a transition **a** leading to a violation state, and one leading to a legal one, will—if they act rationally from a decision-theoretic perspective—chose for the latter.

[9] That is to say: $\mathcal{I}^m\lceil \mathtt{Pr} - \mathtt{Pr}^{\mathtt{pay}} = \mathcal{I}^{m'}\lceil \mathtt{Pr} - \mathtt{Pr}^{\mathtt{pay}}$.

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 (7) results, therefore, in the validity of the following property:

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

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

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 [38] would become relevant.

5 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 $\mathtt{punish}(i)$ and $\mathtt{reward}(i)$. As a result of such an update, the original model m results in a sub-model of m'.

Let $r^m_{\mathbf{a}} : W \longrightarrow 2^W$ be the function associating to each state in W the states reachable by $\mathbf{a}$-transitions.

If $w \in r^m_{\mathtt{Act}}$ & $(m', w) \models \mathtt{turn}(j) \wedge \mathtt{payoff}(i, x)$
then $\exists w' \in W$ such that $wR^{m'}_{\mathtt{reward}(i)}w'$ & $(m', w') \models \mathtt{payoff}(i, y)$ with $x \leq y$
and $\exists w'' \in W$ such that $wR^{m'}_{\mathtt{punish}(i)}w'$ & $(m', w'') \models \mathtt{payoff}(i, y)$ with $y < z$

where z is such that $\exists w \in w \in r^m_{\mathtt{Act}}(m, w) \models \mathtt{viol}(i)) \wedge \mathtt{payoff}(i, x)$. What the definition above states is that the update consists in adding to every dead end in m a trivial game consisting of (at least) a binary choice by enforcer j between punishing or rewarding. 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 swoop 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.

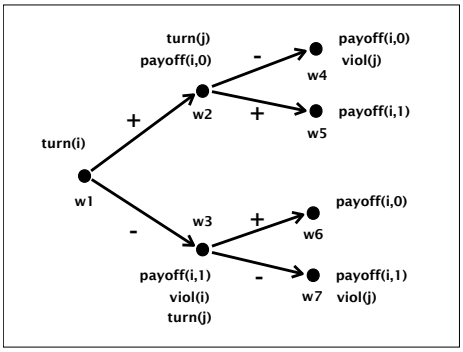

Figure 8: Enforcement norms.

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:

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

$$(\texttt{turn}(j) \wedge \neg\texttt{viol}(i)) \quad \rightarrow \quad [\texttt{punish}(i)]\texttt{viol}(j) \tag{16}$$

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 8). 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.

5.1 Regimenting enforcement norms.

At this point, the norms expressed in Formulae (15) and (16 need implementation. Again, regimentation can be chosen. The result of regimentation of enforcement norms in the running example is depicted in Figure 9. Formally, this corresponds to an update $m \longmapsto m'$ of m where:

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

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

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.

5.2 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

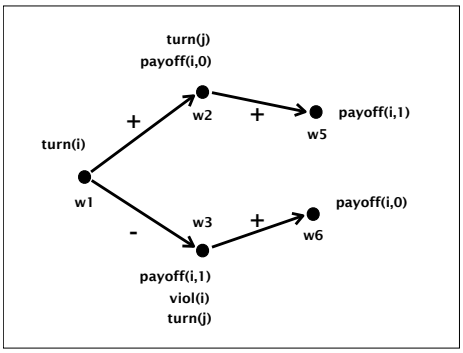

Figure 9: Regimentation of enforcement norms.

autonomously complies with the enforcement norms expressed in Formulae (15) and (16), punishing player i when i commits a violation and rewarding i when i complies (Figure 10). In the running example, perfect enforcement

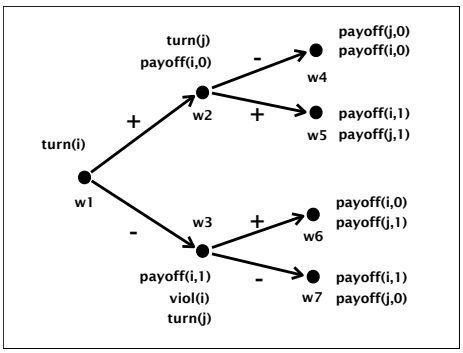

Figure 10: Perfect enforcement.

of enforcement norms can be defined by a simple update $m \longmapsto m'$ of the interpretation functions of the two models such that:

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

which results in a perfect match between higher payoffs and legal behavior. Figure 11 represents, in strategic form, the extensive game depicted in Figure 10 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 [38]. 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 at this level that a number of concepts and techniques could be imported from

Mechanism Design and Implementation Theory [27, 29, 34, 28] to the formal theory of NMAS.

5.3 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, $\mathit{Regiment}(X)$ to denote the set of norms from X which are regimented or automatically enforced, and $\mathit{Enforce}(X)$ to denote the set of norms containing X together with all the norms specifying the enforcement of X ($X \subseteq \mathit{Enforce}(X)$). The implementation of S is the enforcement of the norms in S which are not regimented: $\mathit{Implement}(X) = \mathit{Enforce}(X \setminus \mathit{Regiment}(X))$.

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 $\mathit{Regiment}(X) = X$ then there is no norm left to be implemented. Instead if $\mathit{Regiment}(X) \subset X$ then $\emptyset \subset \mathit{Implement}(S)$, which means that the implementation operation should be iterated on $\mathit{Implement}(X)$. In principle, such iteration is endless, unless there exists a final implementation level whose norms are all regimented or automatically enforced.

6 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 [23] and intermediate concepts [32].

As an example, consider the model m' obtained via the update of the initial model m corresponding to perfect enforcement (Figure 10). 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

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

Figure 11: Enforcement of the Blocks World scenario in strategic form

norms expressed in Formulae (15) and (16) by the following norm:

$$\texttt{turn}(j) \rightarrow [\texttt{reward}(i)]\texttt{viol}(j) \tag{17}$$

and then update m to m'' in order to implement the norm expressed in Formula (17), 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_3, thereby automatically triggering the enforcement norms expressed in Formulae (15) and (16). As a result, the enforcement mechanism in place in model m' are imported "for free" by m'' via simply changing the meaning of $\texttt{viol}(i)$ (Figure 12). As you can see, the payoffs for enforcer j are different from Figure 10.

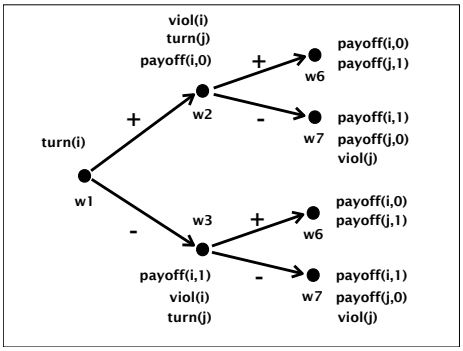

Figure 12: Implementation via norm change.

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

$$\texttt{on}(\texttt{b},\texttt{a}) \wedge \texttt{clear}(\texttt{b}) \wedge \texttt{clear}(\texttt{c}) \wedge \texttt{turn}(i) \quad \rightarrow \quad [\texttt{move}(\texttt{b},\texttt{a},\texttt{c})(i)]\texttt{viol}(i) \tag{18}$$

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 (15) and (16).

7 Related work

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

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

Since BDI-CTL [17] 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 [15, 16]. 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.

Another obvious candidate for the question mark is a theory of normative systems [1]. Whereas deontic logic assumes a normative system such as a legal code in the background which is not made explicit, a normative system makes the norms themselves explicit, such that we can say that a norm is active, in force, violated, and so on [47]. See [25] for an up to date review on the distinction between a theory of norative 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 multiagent structure of a normative system.

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 [43, 44, 45] 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.

The second candidate is Boella and van der Torre's game-theoretic approach to normative multiagent systems, which studies the more general problem of norm creation [7, 12, 11]. 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 [8]. They focus in particular on the enforcement of norms using enforcers, and discuss the role of procedural norms to motivate the enforcers [9]. 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 [8]. 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 [11] for a detailed discussion on their approach.

There are many organizational and institutional theories, such as the ones proposed in [23], and there is a lot of work on coordination and the environment [20, 40]. 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.

8 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 en-

forcers, and implementation via norm change. The 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.

Davide Grossi is supported 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 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.

[8] G. Boella and L. van der Torre. Enforceable social laws. In *Procs. of 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)*, pages 682–689, New York (NJ), 2005. ACM Press.

[9] G. Boella and L. van der Torre. Substantive and procedural norms in normative multiagent systems. *Journal of Applied Logic*, in press.

[10] 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.

[11] Guido Boella and Leendert van der Torre. A game-theoretic approach to normative multi-agent systems. In Guido Boella, Leon van der Torre, and Harko Verhagen, editors, *Normative Multi-agent Systems*, number 07122 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.

[12] Guido Boella and Leendert van der Torre. A game-theoretic approach to normative multi-agent systems. In *Normative Multi-agent Systems (NorMAS'07)*, in press.

[13] Guido Boella, Leon van der Torre, and Harko Verhagen, editors. *Normative Multi-Agent Systems*, number 07122 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.

[14] W. Briggs and D. Cook. Flexible social laws. In *Proceedings 14th International Joint Conference on Artificial Intelligence*, pages 688–693, 1995.

[15] 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.

[16] Jan Broersen, Mehdi Dastani, and Leendert van der Torre. Bdioctl: Obligations and the specification of agent behavior. In *Proceedings of IJCAI'03*, pages 1389–1390, 2003.

[17] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

[18] Rosaria Conte, Cristiano Castelfranchi, and Frank Dignum. Autonomous norm acceptance. In Jörg Müller, Munindar P. Singh, and Anand 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.

[19] D. D. Harel amd 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.

[20] Mehdi Dastani, Farhad Arbab, and Frank S. de Boer. Coordination and composition in multi-agent systems. In *Procs. of AAMAS*, pages 439–446, 2005.

[21] D. Fitoussi and M. Tennenholtz. Choosing social laws for multi-agent systems: Minimality and simplicity. *Artificial Intel ligence*, 119:61–101, 2000.

[22] 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 multiagent systems*, pages 667–673. ACM Press, 2005.

[23] D. Grossi. *Designing Invisible Handcuffs. Formal Investigations in Institutions and Organizations for Multi-agent Systems*. PhD thesis, Utrecht University, SIKS, 2007.

[24] 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.

[25] Joerg Hansen. *Imperatives and Deontic Logic: On the Semantic Foundations of Deontic Logic*. University of Leipzig, 2008.

[26] W. van der Hoek, M. Roberts, and M. Wooldridge. Social laws in alternating time: Effectiveness, feasibility, and synthesis. *Synthese*, 156(1), 2007.

[27] 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.

[28] M. O. Jackson. A crash course in implementation theory. *Social Choice and Welfare*, 18:655–708, 2001.

[29] M. O. Jackson. Mechanism theory. In U. Derigs, editor, *Encyclopedia of Life Support Systems*. EOLSS Publishers, 2003.

[30] A. J. I. Jones and M. Sergot. On the characterization of law and computer systems. *Deontic Logic in Computer Science*, pages 275–307, 1993.

[31] S. Kanger. New fondations for ethical theory. In R. Hilpinen, editor, *Deontic Logic: Introductory and Systematic Readings*, pages 36–58. Reidel Publishing Company, 1971.

[32] 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.

[33] F. Lopez, M. Luck, and M. d'Inverno. A normative framework for agent-based systems. *Computational and Mathematical Organization Theory*, 12:227–250, 2006.

[34] E. Maskin. Nash equilibrium and welfare optimality. *Review of Economic Studies*, 66:23–38, 1999.

[35] 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.

[36] D. C. North. *Institutions, Institutional Change and Economic Performance*. Cambridge University Press, Cambridge, 1990.

[37] S. Onn and M. Tennenholtz. Determination of social laws for multi-agent mobilization. *Artificial Intel ligence*, 95:155–167, 1997.

[38] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[39] 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.

[40] A. Ricci, A. Omicini, and E. Denti. Activity theory as a framework for mas coordination. In *Procs. of ESAW'02*, pages 96–110, 2002.

[41] S. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach*. Prentice Hall International, 2001.

[42] 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.

[43] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73(1-2):231–252, 1995.

[44] Y. Shoham and M. Tennenholtz. On the emergence of social conventions: Modeling, analysis and simulations. *Artificial Intelligence*, 94(1–2):139–166, 1997.

[45] M. Tennenholtz. On stable social laws and qualitative equilibria. *Artificial Intelligence*, 102(1):1–20, 1998.

[46] M. Tennenholtz. On social constraints for rational agents. *Computational Intelligence*, 15(4), 1999.

[47] L. van der Torre and Y. Tan. Diagnosis and decision making in normative reasoning. *Artificial Intelligence and Law*, 7(1):51–67, 1999.