Verifying Normative Behaviour via Normative Mechanism Design

Nils Bulling
Department of Informatics,
Clausthal University of Technology, Germany
bulling@in.tu-clausthal.de

Mehdi Dastani
Intelligent Systems Group,
Utrecht University, The Netherlands
mehdi@cs.uu.nl

Abstract

The environment is an essential component of multi-agent systems and is often used to coordinate the behaviour of individual agents. Recently many languages have been proposed to specify and implement multi-agent environments in terms of social and normative concepts. In this paper, we first introduce a formal setting of multi-agent environment which abstracts from concrete specification languages. We extend this formal setting with norms and sanctions and show how concepts from mechanism design can be used to formally analyse and verify whether specific normative behaviours can be enforced (or implemented) if agents follow their subjective preferences. We also consider complexity issues of associated problems.

1 Introduction

The overall objectives of multi-agent systems can be ensured by coordinating the behaviors of individual agents and their interactions. Existing approaches advocate the use of exogenous normative environments and organisational models to regulate the agents' behaviors and interactions [5; 6; 7; 11]. Norm-based environments regulate the behavior of individual agents in terms of norms being enforced by means of regimentation or sanctioning mechanisms [5; 6]. Generally speaking, the social and normative perspective is conceived as a way to make the development and maintenance of multi-agent systems easier to manage.

This paper aims at providing a formal analysis of rule-based normative environments from the mechanism design perspective. In particular, an environment in a multi-agent system is modelled as a concurrent game structure (which is also considered as mechanism) where possible paths in the game structure denote possible execution traces of the corresponding multi-agent environment. Adding norms to the environment may change its set of possible execution traces which in turn correspond to the set of paths of an update of the concurrent game structure. The correspondence between (normative) multi-agent environment and (updated) game structure is an attempt to bridge the gap between normative environments and mechanism design. This relation sets the stage for studying formal properties of normative environments such as whether a set of norms implements specific choice functions in specific equilibria. This also allows, for example, to analyse whether groups of agents are willing to obey the rules specified by a normative system. The formal analysis is closely related to work presented in [1; 16] where norms are modelled by the deactivation of transitions. Adding norms to environments in our model may, however, either deactivate transitions or on the contrary add new transitions. Our work is also motivated by [14; 15] where social laws were proposed to be used in computer science to control agents' behaviours.

This paper is motivated by 2OPL (Organisation Oriented Programming Language), which is a rule-based language to implement normative environments in terms of norms and sanctions [5; 6]. In particular, it provides constructs to specify 1) the (initial) state of an environment, 2) the outcomes of agents' actions, and 3) norms and sanctions. The execution of a normative environment specification, with a sequence of agents' synchronized actions as input, determines the normative outcomes of the action sequence. Although many other, even more expressive, languages have been proposed to specify normative environments, 2OPL has the advantage that it comes with a full operational semantics and computationally attractive complexity. This allows normative environments to be studied and analysed as formal mechanisms.

The structure of this paper is as follows. First, we explain our notion of normative environment for multi-agent systems. We propose a formal setting for multi-agent environments and show how they can be related to concurrent game structures and mechanism design. Then, we introduce a specification language for normative multi-agent environments and extend the formal setting of environments with norms and sanctions. Finally, we study and analyse formal properties of normative multi-agent environments using their corresponding concurrent game structure and concepts from mechanism design. We prove the complexity of two verification problems.

2 Formal Framework

2.1 Motivation: Multi-Agent Systems

A multi-agent system consists of a set of agents and an environment in which the agents perform their actions. We assume that agents may or may not be aware of the environment specification, perform only synchronous actions in the environment, and that the outcome of agents' synchronized actions is determined by the environment. In this setting, an environment is specified in terms of an initial state and a set of synchronized action specifications. Agents can change the state of the environment by performing synchronized actions.

Following [5; 6], a *normative environment* is specified in terms of an *initial state*, a set of *synchronized action specifications*, and two sets of rules that represent *norms* and *sanctions*. A normative environment determines the outcome of actions by means of action specifications together with norms and sanctions. Norms are represented by *counts-as rules* [10]. A counts-as rule indicates that a specific state (satisfying the rule's antecedent) counts as a violated state (represented by a specific violation proposition). For example, a traffic norm can be represented by the counts-as rule $\{carParked, \neg Pay\} \Rightarrow \{violPark\}$ stating that parking the car without paying ($carParked$ and $\neg Pay$) counts as a parking violation ($violPark$). Sanctions are also represented by rules [5]. A *sanction rule* connects violation propositions (the rule's antecedent) to state propositions (the rule's consequent). The state propositions in sanction rules denote sanctions and specify how a violated environment state should be repaired and turned back to an optimal state. For example, a sanction rule $\{violPark\} \Rightarrow \{fine50\}$ connects the parking violation to a state proposition denoting a fine of 50 euro being issued. A pair consisting of a set of counts-as rules and a set of sanction rules is called a *norm set*.

A normative environment is thus an environment together with a norm set. We assume the sets of counts-as and sanctions rules to be finite.

2.2 Notation: Propositions, States, Actions

We assume that Π is a finite and fixed set of *propositional symbols*. For each $X \subseteq \Pi$, we use X^- to refer to $\{\neg \mathsf{p} \mid \mathsf{p} \in X\}$, $X^c = X \cup (\Pi \setminus X)^-$ (closed-world assumption for X), and $X^l = X \cup X^-$ (literals). Given a set $X \subseteq \Pi$ and a consistent set $Y \subseteq \Pi^l$ we define the *update* of X by Y as $X \oplus Y = \{\mathsf{p} \in \Pi \mid \mathsf{p} \in X \cup Y \text{ and } \neg\mathsf{p} \notin Y\}$. Note that $X \oplus Y$ consists of all propositional atoms from Y and extends them with atoms from X as long as they are consistent with (the negated atoms from) Y. We assume the reader is familiar with standard propositional inference, e.g. $X \models^{\mathsf{PL}} \bigwedge Y$ means that formula $\bigwedge Y$ ($\bigwedge Y$ stands for $\bigwedge_{l \in Y} l$) is a consequence of theory X using the *propositional calculus* where $X, Y \subseteq \Pi^l$. A set $X \subseteq \Pi^l$ is said to be *consistent* iff $X \not\models^{\mathsf{PL}} \bot$.

In the following we use Q to denote a set of *states*. We assume that states are given by propositional facts, that is $Q \subseteq \mathcal{P}(\Pi)$. Note, that $|Q| \leq 2^{|\Pi|}$. We chose this representation due to the following specific notion of action profile specification. We assume that $\mathbb{A}\text{gt} = \{1, 2, \ldots, k\}$ is a finite, fixed and non-empty set of *agents*. We typically use $i, i_1, i_2 \ldots$ as variables for agent names. Each agent $i \in \mathbb{A}\text{gt}$ is assumed to have at its disposal a set of (individual) actions that it can perform in the environment. A (synchronous) action profile consists of an action for each agent $i \in \mathbb{A}\text{gt}$. The *specification* of an *action profile* $\vec{\alpha} = (\alpha_1, \ldots, \alpha_k)$ in an environment, where α_i is the individual action of agent i, is given by $(P, \vec{\alpha}, E)$ where $P, E \subseteq \Pi^l$ are consistent sets. The *precondition* P specifies the applicability of action profile $\vec{\alpha}$. Action profile $\vec{\alpha}$ is *applicable in a state* q iff $q^c \models^{\mathsf{PL}} \bigwedge P$. The *postcondition* E specifies the effect of performing $\vec{\alpha}$ in a state, i.e., how a state changes after the action has been performed. Performing $\vec{\alpha}$ in state q results in state $q \oplus E$. Note that $\oplus$ is a simple update operator that adds/removes propositional atoms to/from states.

2.3 Concurrent Structures and Strategies

In the following we introduce *concurrent (game) structures (*CGS*s)* from [2] (modulo minor modifications). They serve as models for our formal analysis of the environment in multi-agent systems. An environment in a multi-agent system is assumed to be specified in terms of a set of states, among which an initial state, and a set of (synchronized) action specifications. Informally speaking, a CGS is given by a labelled transition system where transitions are activated by action profiles. In our setting we construct a CGS from a given set Q of states, an initial state $q^I \in Q$, and a set $ActSpec$ of action specifications. A CGS represents possible evolutions of an environment as a consequence of agents performing (synchronized) action profiles. Formally, we define the $(ActSpec, Q, q^I)$*-generated* CGS as the CGS which is obtained by applying all action profiles to the set of states.

Essentially, these models can be seen as *mechanisms*. They play the same role as their game theoretic counterparts and define the rules and the structure of the game. Henceforth, we will use the terms CGS and mechanism interchangeably.

Definition 1 (Generated CGS, $\mathfrak{M}$**)** *Let* Π *be a set of atomic propositions,* $ActSpec$ *be a set of action specifications,* $Q \subseteq \mathcal{P}(\Pi)$ *be a set of states, and* $q^I \in Q$*. The* $(ActSpec, Q, q^I)$-generated CGS *is given by* $\mathfrak{M} = \langle \mathbb{A}\text{gt}, Q, Act, \hat{d}, d, o\rangle$ *where*

- $\mathbb{A}\text{gt} = \{1, \ldots, k\}$ *is a nonempty finite set of agents (as before).*
- $Act = \{\alpha_i \mid (P, (\alpha_1, \ldots, \alpha_k), E) \in ActSpec, 1 \leq i \leq k\}$ *is a nonempty finite set of individual actions.*
- $\hat{d} : Q \to \mathcal{P}(Act^k)$ *is a function that assigns a set of applicable action profiles to each state.* $\vec{\alpha} \in \hat{d}(q)$ *iff* $\exists (P, \vec{\alpha}, E) \in ActSpec : q^c \models^{\mathsf{PL}} \bigwedge P$ *and* $q \oplus E \in Q$*. Hence, an action profile is applicable if its precondition is satisfied and if the resulting state belongs to* Q*.*
- $d : \mathbb{A}\text{gt} \times Q \to \mathcal{P}(Act)$ *is a function that assigns nonempty sets of applicable individual actions to each agent at each state, i.e.* $d(i, q) = \{\alpha_i \mid (\alpha_1, \ldots, \alpha_k) \in \hat{d}(q), 1 \leq i \leq k\}$*. We write* $d_i(q)$ *instead of* $d(i, q)$*.*
- $o : Q \times Act^k \to Q$ *is a partial (deterministic) transition function that determines the outcome state of an action profile performed in a state. It is defined on all* $(q, \vec{\alpha})$ *with* $\vec{\alpha} \in \hat{d}(q)$*. We have* $o(q, \vec{\alpha}) = q \oplus E$ *for* $(P, \vec{\alpha}, E)$*. We assume that other action profiles simply fail and cannot be executed.*

If clear from context we will simply use $\mathfrak{M}$ *to denote the* $(ActSpec, Q, q^I)$-generated CGS. *We use* $X_{\mathfrak{M}}$ *to refer to element* X *contained in* $\mathfrak{M}$*, e.g.* $Q_{\mathfrak{M}} = Q$*.*

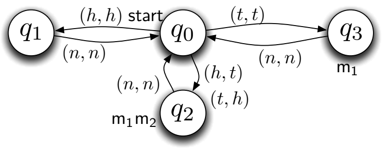

Figure 1: A simple CGS $\mathfrak{M}_0$.

Example 1 *For a set* $\Pi = \{\mathsf{start}, \mathsf{m}_1, \mathsf{m}_2\}$ *of propositions, we define the set* $ActSpec$ *of action profiles as follows:* $ActSpec = \{(\{\mathsf{start}\}, (h,h), \{\neg\mathsf{start}\}), (\{\neg\mathsf{start}\}, (n,n), \{\mathsf{start}, \neg\mathsf{m}_1, \neg\mathsf{m}_2\}), (\{\mathsf{start}\}, (h,t), \{\mathsf{m}_1, \mathsf{m}_2, \neg\mathsf{start}\}), (\{\mathsf{start}\}, (t,h), \{\mathsf{m}_1, \mathsf{m}_2, \neg\mathsf{start}\}), (\{\mathsf{start}\}, (t,t), \{\mathsf{m}_1\}, \neg\mathsf{start})\}$. *Figure 1 shows the* $(ActSpec, \{q_0, q_1, q_2, q_3\}, q_0)$*-generated* CGS $\mathfrak{M}_0$. *For convenience, we identify the label of the node with the set of propositions true in it, e.g.* $q_0 = \{\mathsf{start}\}$. *Agents can execute actions t, h, and n, standing for "show tail", "show head", and "no operation", respectively.*

A *strategy* of agent i is a conditional plan that specifies what i is going to do in each state. In this paper we focus on state-based strategies (due to obvious reasons they are also called memoryless strategies).

Definition 2 (Strategy, valid) *A* (memoryless) strategy *for agent* i *is a function* $s_i : Q \to Act$ *such that* $s_i(q) \in d_i(q)$. *A* collective strategy s_A *for a group* $A = \{i_1, \ldots, i_l\} \subseteq \mathbb{A}\mathrm{gt}$ *of agents is a tuple* $(s_{i_1}, \ldots, s_{i_l})$ *of individual strategies, one per agent in A. For* $A = \mathbb{A}\mathrm{gt}$ *we obtain a* (complete) strategy profile. *Given two collective strategies* s_A *and* s_B, $A \cap B = \emptyset$, *we write* (s_A, s_B) *to denote the collective strategy of* $A \cup B$.

We say that a complete strategy profile s *is* valid *iff for all* $q \in Q$ *it holds that* $s(q) \in \hat{d}(q)$.

A *path* $\lambda = q_0 q_1 q_2 \ldots$ is an infinite sequence of states such that there is a transition between each q_m, q_{m+1}. We use $\lambda[m]$ to denote the mth position on path λ (starting from $m = 0$) and $\lambda[m, \infty]$ to denote the subpath of λ starting from m.

Definition 3 (Outcome) *The* outcome $out_{\mathfrak{M}}(q, s)$ of a valid strategy profile s from state q in model $\mathfrak{M}$ *is the (unique) path* $\lambda = q_0 q_1 q_2 \ldots$ *such that* $q_0 = q$ *and* $q_m = o(q_{m-1}, s(q_{m-1}))$ *for each* $m = 1, 2, \ldots$.

2.4 Agent's Preferences

In the analysis of multi-agent systems preferences of agents are often of utmost importance. They are the driving force of agents' behaviours. In our considered models, the executions of the environment and thus the corresponding CGS-paths can be seen as the semantics of preferences. Hence, it is a natural consequence to assume that agents prefer some executions over others. We do not directly consider sets of paths to denote preferences (they may not be regular), but rather temporal formulae. They are used to describe sets of paths, namely all those paths which satisfy them. This idea was already followed in [1; 16; 4] where **CTL** and **ATL**, respectively, have been used to model agents' preferences.

Here, we use the temporal logic $\mathcal{L}_{LTL}$ [13] for modelling preferences of agents. The logic extends propositional logic with operators that allow to express temporal patterns over infinite sequences of states, called *paths*. The basic temporal operators are $\mathcal{U}$ (*until*), $\Box$ (*always*), $\Diamond$ (*eventually*) and $\bigcirc$ (*in the next state*). The *language* $\mathcal{L}_{LTL}(\Pi)$ is given by all formulae generated by the following grammar, where $\mathsf{p} \in \Pi$ is a proposition: $\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi\mathcal{U}\varphi \mid \bigcirc\varphi \mid \Box\varphi \mid \Diamond\varphi$. *Models* for this language are defined as ω-sequences (infinite paths) λ over $\mathcal{P}(\Pi)$, i.e. $\lambda \in \mathcal{P}(\Pi)^\omega$. We assume the reader is familiar with the *semantics* of **LTL**. We note that in our setting we will not make use of the full expressivity of the logic **LTL** and its accompanied high computational complexity (cf. Def. 9) which matches the computationally attractive properties of the (restricted) language used for norms and sanctions.

3 Normative Mechanism Design

Here we introduce *normative mechanism design* by considering a mechanism with norms, sanctions and preferences.

3.1 Basic Setting and Definitions

In social choice theory a *social choice function* assigns outcomes to given profiles of preferences (see e.g. [12]). Various natural requirements are imposed on social choice functions in order to ensure e.g. fairness. *Mechanism design* is concerned with creating a protocol/a set of rules such that the outcome agrees with a social choice function f provided that agents behave rationally–in some sense–according to their preferences. In game theoretic terms *behaving rationally* means to act according to some *solution concept* $\mathcal{S}$ (e.g. the concept of Nash equilibria). If such a mechanism exists it is said that it $\mathcal{S}$*-implements* f.

We define a *normative behaviour function* as a function that assigns a set of "desired" environment executions (represented by $\mathcal{L}_{LTL}$-formulae) to each preference profile (a sequence of sequences of $\mathcal{L}_{LTL}$-formulae) of the agents. We refer to the outcome as the *normative outcome wrt a specific preference profile*. In our view, the aim of *normative mechanism design* is to come up with a *normative mechanism* (i.e., an environment specified in terms of actions, norms and sanctions) such that the agents–again following some rationality criterion according to their preferences–behave in such a way that the environment executions stay within the normative outcome. The idea is that norms and sanctions will (de)motivate agents performing specific actions.

We begin with introducing the set of preferences and outcomes defined over elements used in a CGS and in $\mathcal{L}_{LTL}$.

Definition 4 (Preference) *A* preference list *of an agent* $i \in \mathbb{A}\mathrm{gt}$ *is given by a finite sequence* $\gamma_i = ((\varphi_1, u_1), \ldots, (\varphi_{l-1}, u_{l-1}), (\varphi_l, u_l))$ *where each* $\varphi_j \in \mathcal{L}_{LTL}(\Pi)$, *each* $u_j \in \mathbb{N}_0$ *for* $j = 1, \ldots, l-1$, $\varphi_l = \top$, *and* $u_l = 0$. *Intuitively, a path* λ *is assigned utility* u_j *if* φ_j *is the first formula in the list which is satisfied on* λ.

A preference (profile) *is given by* $\vec{\gamma} = (\gamma_1, \ldots, \gamma_k)$ *containing a preference list for each agent. We typically use* $\mathcal{P}refs$ *to denote a non-empty set of such preferences.*

The definition of a normative behaviour function is straightforward given the explanation above.

Definition 5 (Normative behaviour function)
A $(\mathfrak{M}, \mathcal{P}refs)$-based normative behaviour function f *is a mapping* $f : \mathcal{P}refs \to \mathcal{L}_{LTL}(\Pi)$.

We have argued above that a normative mechanism may be considered as an environment together with a set of norms and sanctions that guide agents' behaviours in certain directions. Agents are assumed to behave according to their preferences. As the fulfilment of these preferences does only depend on the valuation of states on a path, agents' behavior may be effected if imposing norms and sanctions would cause a modification of states' valuations.

In this paper, we follow [5] and represent norms as a set of counts-as rules and sanctions as a set of sanction rules. The advantages of this approach are the existence of an operational semantics (already implemented), its simplicity (rule based representation and computational complexity), and its expressiveness (p counts-as a violation can be interpreted as it is obligatory that $\neg p$ or p is forbidden). A counts-as rule indicates that a state satisfying its antecedent counts as a state with a specific violation. An example is given in Section 2.1.

Definition 6 (Norm Set M, $R^{\mathbf{cr}}$, $R^{\mathbf{sr}}$) *A* norm set M over propositions Π *is given by* $M = (\mathsf{Viol}, R^{cr}, R^{sr})$ *where* Viol *is a set of propositions disjoint with* Π *indicating* norm violations*,* $R^{cr} \subseteq \mathcal{P}(\Pi^l \cup \mathsf{Viol}) \times \mathcal{P}(\mathsf{Viol})$ *is a set of* count-as rules*, and* $R^{sr} \subseteq \mathcal{P}(\mathsf{Viol}) \times \mathcal{P}(\Pi^l)$ *a set of* sanction rules. *For a rule $r = B \times H \in R^{cr} \cup R^{sr}$ we also write $B \Rightarrow H$ and use* $\mathsf{body}(r)$ *(resp.* $\mathsf{head}(r)$*) to refer to B (resp. H). The* empty norm set *is defined as $M_\epsilon = (\emptyset, \emptyset, \emptyset)$.*

The basic idea is that these norms imposed on a model change the states' valuation. The set of counts-as rules *applicable* wrt a set of propositions $X \subseteq \Pi$ is defined as follows:

$$\mathsf{App}^{cr}(X, R^{\mathrm{cr}}) = \{r \in R^{\mathrm{cr}} \mid X^c \models^{\mathsf{PL}} \bigwedge \mathsf{body}(r)\}.$$

Next, we use function $\mathsf{cl}_X^{R^{\mathrm{cr}}}(Y) = \{\mathsf{head}(r) \mid r \in \mathsf{App}^{cr}(X \cup Y, R^{\mathrm{cr}})\}$ and define $\mathsf{cl}_X^{R^{\mathrm{cr}}}\uparrow^{\omega}$ as the *smallest fixed-point* of $\mathsf{cl}_X^{R^{\mathrm{cr}}}(\cdot)$, which exists due to *Knaster/Tarski's fixed point theorem*. This fixed point provides the set of heads of all applicable counts-as rules, i.e. the set of all violation propositions wrt to a set of propositions X. Finally, we define the result of applying the norm set M to a state $q \in Q$ as follows:

$$q \upharpoonright M := q \oplus \{\mathsf{head}(r) \mid (\mathsf{cl}_q^{R^{\mathrm{cr}}}\uparrow^{\omega})^c \models^{\mathsf{PL}} \mathsf{body}(r) \text{ and } r \in R^{\mathrm{sr}}\}.$$

The resulting state is a state in which all violations are detected and corresponding sanctions are imposed. Note that imposing sanctions modifies the state. Consequently, we define the result of applying the norm set M to a set of states Q as follows: $Q \upharpoonright M = \{q \upharpoonright M \mid q \in Q\}$.

That is, we take each state $q \in Q$, determine all applicable count-as rules to q (until a fixed point is reached) and generate the violation propositions, derive the sanctions, and update q with the derived sanctions.

Definition 7 ($\mathfrak{M} \upharpoonright M$, normative mechanism) *Let $\mathfrak{M}$ be the $(ActSpec, Q, q^I)$-generated* CGS. *The* normative update *of $\mathfrak{M}$ with the norm set M, denoted $\mathfrak{M} \upharpoonright M$, is defined as the $(ActSpec, Q \upharpoonright M, q^I \upharpoonright M)$-generated* CGS. *We call the tuple $(\mathfrak{M}, M)$ a* normative mechanism.

We would like to point out that $\mathfrak{M} \upharpoonright M$ is a mechanism whereas $(\mathfrak{M}, M)$ is a *normative* mechanism.

Note that after a normative update took place some action specifications may become inapplicable where others become applicable, so we remove *and* create transitions which is an important difference to [1].

3.2 Implementation of Normative Mechanisms

This section examines the question of whether specific outcomes of a normative mechanism agree with a normative behaviour function. In this setting it is assumed that agents behave rationally in some way (they follow some solution concept). In order to use these game theoretic concepts we link our setting to strategic games.

Definition 8 (CGS $\rightsquigarrow$ strategic game) *Let $\mathfrak{M}$ be a* CGS*, $k = |\mathbb{A}\mathrm{gt}_{\mathfrak{M}}|$, $\vec{\gamma} = (\gamma_1, \dots, \gamma_k) \in \mathcal{P}refs$ be a preference profile, and q^I be the initial state in $\mathfrak{M}$.*

We define $\Gamma(\mathfrak{M}, \vec{\gamma}, q^I)$, the strategic game associated with *$\mathfrak{M}, \vec{\gamma}$, as the strategic game $\langle \mathbb{A}\mathrm{gt}_{\mathfrak{M}}, \Sigma_1, \dots, \Sigma_k, \mu\rangle$ (cf. [12]) where $\mathbb{A}\mathrm{gt}_{\mathfrak{M}}$ is the set of players, Σ_i is the set of strategies of player i, and the payoff function $\mu = (\mu_1, \dots, \mu_k)$ is defined as follows:*

$$\mu_i(s) = \begin{cases} u_j & \textit{if } s \textit{ is a complete valid strategy profile,} \\ -1 & \textit{else,} \end{cases}$$

where $1 \le i \le k$, $\gamma_i = ((\varphi_1, u_1), \dots, (\varphi_{l-1}, u_{l-1}), (\varphi_l, u_l))$ and j is the minimal index such that $out_{\mathfrak{M}}(q^I, s) \models^{\mathsf{LTL}} \varphi_j$. (We note that μ is well-defined as $\varphi_l \equiv \top$).

A game theoretical solution concept, e.g. Nash equilibria or dominant strategies, can be considered as a function $\mathcal{S}$ whose domain is the set of strategic games and its image is a set of strategy profiles. That is, for each strategic game $G = (\mathbb{A}\mathrm{gt}, (\Sigma_i), \mu)$ we have that $\mathcal{S}(G) \subseteq \times_{i=1,\dots,|\mathbb{A}\mathrm{gt}|}\Sigma_i$.

We assume the reader is familiar with the solution concept of Nash equilibria and dominant strategy equilibria and refer to [12] for more details. We use $\mathcal{NE}$ and $\mathcal{DOE}$ to refer to these concepts, respectively.

We lift the notion of solution concepts to a CGS $\mathfrak{M}$ by defining $\mathcal{S}(\mathfrak{M}, \vec{\gamma}, q) := \mathcal{S}(\Gamma(\mathfrak{M}, \vec{\gamma}, q))$.

Definition 9 (M $\mathcal{S}$-implements f over $\mathfrak{M}, q^I, \mathcal{P}refs$) *We say that a norm set M $\mathcal{S}$-*implements *normative* behaviour function *f over $\mathfrak{M}, q^I, \mathcal{P}refs$ iff the following condition is satisfied for all $\vec{\gamma} \in \mathcal{P}refs$ and all $\forall s \in \mathcal{S}(\mathfrak{M} \upharpoonright M, \vec{\gamma}, q^I) : out_{\mathfrak{M}\upharpoonright M}(q^I, s) \models^{\mathsf{LTL}} f(\vec{\gamma})$. If there is some M that $\mathcal{S}$-implements f over $\mathfrak{M}, q^I, \mathcal{P}refs$, we say that f is $\mathcal{S}$-*implementable over *$\mathfrak{M}, q^I, \mathcal{P}refs$.*

We highlight that our notion of implementation is given wrt. a norm set and not wrt. a mechanism per se.

Example 2 *We consider the* CGS *$\mathfrak{M}$ from Example 1 with $q^I = q_0$. By strategy s_x for $x \in \{t, h\}$ we denote the strategy with $s_x(q_0) = x$ and $s_x(q) = n$ for all other states $q \neq q_0$. We define $\mathcal{P}refs = \{\gamma^1, \gamma^2\}$ with $\gamma^1 = (((\Diamond(\mathsf{m}_1 \wedge \mathsf{m}_2), 1), (\top, 0)), ((\Diamond \mathsf{m}_2, 1), (\top, 0)))$ and $\gamma^2 = (((\Box\neg\mathsf{m}_1, 1), (\top, 0)), (\Diamond(\mathsf{m}_1 \vee \mathsf{m}_2 \wedge \Box\neg(\mathsf{m}_1 \wedge \mathsf{m}_2, 1), (\top, 0))))$. Let the following normative behaviour function f be given: $f(\gamma^i) = \Box\neg(\mathsf{m}_1 \wedge \mathsf{m}_2)$ for $i \in \{1, 2\}$. First of all we note that the (empty) norm set $M_\emptyset$ does not $\mathcal{NE}$-implement f over $\mathfrak{M}, q_0, \mathcal{P}refs$ because $\mathcal{NE}(\mathfrak{M}, \gamma^1, q_0) = \{(s_t, s_h), (s_h, s_t)\}$ but $f(\gamma^1)$ is false on*

$\Gamma(\mathfrak{M}, \gamma^1, q_0)$:

1/2	s_h	s_t
s_h	$0,0$	$\mathbf{1,1}$
s_t	$\mathbf{1,1}$	$0,0$

$\Gamma(\mathfrak{M}, \gamma^2, q_0)$:

1/2	s_h	s_t
s_h	$\mathbf{1,0}$	$0,0$
s_t	$0,0$	$\mathbf{0,1}$

$\Gamma(\mathfrak{M} \upharpoonright M, \gamma^1, q_0)$:

1/2	s_h	s_t
s_h	$\mathbf{0,0}$	$-1,-1$
s_t	$-1,-1$	$\mathbf{0,0}$

$\Gamma(\mathfrak{M} \upharpoonright M, \gamma^2, q_0)$:

1/2	s_h	s_t
s_h	$\mathbf{1,1}$	$0,0$
s_t	$0,0$	$\mathbf{0,1}$

Figure 2: Strategic games for S, γ^1, γ^2, and M.

both paths resulting from these profiles. (We recall that $\mathfrak{M} \upharpoonright M_\emptyset = \mathfrak{M}$.)

Now, consider the norm set $M = (\{\mathsf{v}\}, \{\{\neg\mathsf{start}, \mathsf{m}_1, \mathsf{m}_2\} \Rightarrow \{\mathsf{v}\}\}, \{\mathsf{v}\} \Rightarrow \{\neg\mathsf{m}_1\}\}\})$. The mechanism $\mathfrak{M} \upharpoonright M$ is similar to $\mathfrak{M}$ but state $q_2 = \{\mathsf{m}_2\}$. In particular, both action profiles (h, t) and (t, h) are not applicable anymore. Then, M $\mathcal{NE}$-implements f over $\mathfrak{M}, q_0, \mathcal{P}refs$. In Figure 2 we have shown the strategic games $\Gamma(\mathfrak{M}, \gamma^i, q_0)$ and $\Gamma(\mathfrak{M} \upharpoonright M, \gamma^i, q_0)$ for $i = 1, 2$. The Nash equilibria in each game are highlighted in bold letters. In the game $\Gamma(\mathfrak{M} \upharpoonright M, \gamma^2, q_0)$ we have that $\mathcal{NE}(\mathfrak{M} \upharpoonright M, \gamma^2, q_0) = \{(s_h, s_h), (s_t, s_t)\}$, $out_{\mathfrak{M}\upharpoonright M}(q, (s_h, s_h)) = \{(q_0q_1)^\omega\}$, and $out_{\mathfrak{M}\upharpoonright M}(q, (s_t, s_t)) = \{(q_0q_3)^\omega\}$. For each path λ in the outcome we have $\lambda \models^{\mathsf{LTL}} f(\gamma^1)$. The same holds for γ^2.

4 Verification and Complexity Issues

In this section we consider the complexity of the problem whether some norm set implements some normative behavior function and the complexity needed to determine whether such a norm set exists at all. In this paper we focus on Nash equilibrium implementability. These results are important in order to check whether a normative environment ensures desired behaviors or whether some norm set can be constructed to enforce such behaviors. Due to space limitations we can only present the basic setting and the main results.

Encoding. How do we measure the input? Often, the number of action specification/transitions is exponential in the number of states but could be expressed in a more compact form. A *compact action specification* is given by (P, ξ, e) where P is the precondition defined as before, ξ indicates which actions are applicable (explained below), and e is a list $((\xi_1, S_1), \dots, (\xi_h, S_h))$ where each $S_k \subseteq \Pi^l$ is consistent, $1 \le k \le h$. Each formula $\psi \in \{\xi, \xi_1, \dots \xi_h\}$ is a Boolean formula over propositions exec^i_α where $i \in \mathbb{A}\mathrm{gt}$ and $\alpha \in Act$. We require that $\models^{\mathsf{PL}} \xi(\vec{\alpha}) \to \xi_k(\vec{\alpha})$ for some k where $\vec{\alpha} = (\alpha_1, \dots, \alpha_k)$ is an action profile and $\psi(\vec{\alpha})$ is used to refer to the Boolean formula over $\{\top, \bot\}$ obtained by replacing each exec^i_α with $\top$ (resp. $\bot$) if $\alpha_i = \alpha$ (resp. $\alpha_i \neq \alpha$). The *semantics of* e is defined as $[\![e(\vec{\alpha})]\!] := S_j$ where j, $1 \le j \le h$, is the minimal index such that $\models^{\mathsf{PL}} \xi_j(\vec{\alpha})$. Therefore, (P, ξ, e) gives rise to (several) action specifications $(P, \vec{\alpha}, [\![e(\vec{\alpha})]\!])$ if $\models^{\mathsf{PL}} \xi(\vec{\alpha})$. We use $Act\hat{S}pec$ to denote the *set of compact action specifications* and $ActSpec$ to refer to its associated set of (non-compact) action specifications.

Problems. In this chapter we consider the $(\mathcal{S}, M)$-*implementation problem* $\mathsf{IP}^{\mathcal{S}}_{\mathsf{M}}$ which is given by all $P = (\Pi, Q, Act\hat{S}pec, q^I, f, \mathcal{P}refs, M)$ such that the norm set M $\mathcal{S}$-implements f over $\mathfrak{M}, q^I, \mathcal{P}refs$. (Recall, that $\mathfrak{M}$ is the $(ActSpec, Q, q^I)$-generated CGS.) The $\mathcal{S}$-*implementation problem* $\mathsf{IP}^{\mathcal{S}}$ contains all $P' = (\Pi, Q, Act\hat{S}pec, q^I, f, \mathcal{P}refs)$ such that there is a norm set M with $(\Pi, Q, Act\hat{S}pec, q^I, f, \mathcal{P}refs, M) \in \mathsf{IP}^{\mathcal{S}}_{\mathsf{M}}$. We assume that f is computable in polynomial deterministic time wrt. the size of the input. The *size* of Q and Π is defined as the number of their elements. The size of a compact action specification is $|(P, \xi, e)| = |P| + |\varphi| + \sum_{i=1}^{h} |\xi_i| + |S_i|$. $\mathcal{P}refs$ and M are measured in the canonic way.

The following result holds due to the restricted syntax of the language used to define norm sets: fixed points can be calculated in polynomial deterministic time.

Proposition 1 *Given $Act\hat{S}pec$, Q, and q^I, we can compute $Q \upharpoonright M$, and $q^I \upharpoonright M$ in polynomial deterministic time.*

Theorem 2 *The problem $\mathsf{IP}^{\mathcal{NE}}_{\mathsf{M}}$ is $\mathbf{\Pi^P_2}$-complete.*

Proof. [Idea] Hardness is shown by a reduction of **2QBF**, a well known $\mathbf{\Sigma^P_2}$-complete problem [8], to the complement of $\mathsf{IP}^{\mathcal{NE}}_{\mathsf{M}}$. In [3] it is shown that a **2QBF**-formula $\phi \equiv \exists x_1, \dots, x_m \forall x_{m+1}, \dots, x_n \varphi(x_1, \dots, x_n)$ is true iff a player $\mathbf{v}$ (the verifier, controlling variables $x_1, \dots, x_m$) has a winning strategy $s_\mathbf{v}$ (fixing a valuation for variables $x_1 \dots x_m$) against each strategy $s_\mathbf{r}$ (fixing a valuation for variables $x_{m+1} \dots x_n$) of the refuter $\mathbf{r}$ (controlling variables $x_m, \dots, x_n$) to ensure some **LTL**-formula (in [3] **ATL** is used but it can also be formulated in terms of **LTL**).

The new part in our proof is to construct a preference list, a normative behaviour function, and a normative mechanism such that a "witnessing" strategy $s_\mathbf{v}$ of $\mathbf{v}$ (see above) exists iff there is a Nash equilibrium which falsifies f given the other ingredients (this is exactly the complement of $\mathcal{NE}$-implementability). Our proof is also motivated by [9].

Membership is shown by guessing a strategy profile $s = (s_1, \dots, s_k)$ and checking whether any agent i can improve its outcome by deviating from s_i (again, s_i is non-deterministically guessed). If this is not the case s is a Nash equilibrium and all other steps can be done in polynomial deterministic time. This shows that the complement of $\mathsf{IP}^{\mathcal{NE}}_{\mathsf{M}}$ can be solved in $\mathbf{\Sigma^P_2}$. ■

Theorem 3 *The problem $\mathsf{IP}^{\mathcal{NE}}$ is $\mathbf{\Sigma^P_3}$-complete.*

Proof. [Idea] In order to prove this result we extend the idea from Theorem 2. We reduce a complemented **3QBF** formula $\phi \equiv \forall y_1, \dots, y_l \exists x_1, \dots, x_m \forall x_{m+1}, \dots, x_n \varphi(y_1, \dots, y_l, x_1, \dots, x_n)$ (this problem is well known to be $\mathbf{\Pi^P_3}$-complete [8]) to the complement of an $\mathsf{IP}^{\mathcal{NE}}$ instance. The idea is to use a norm set to generate truth valuations of the universally quantified variables $y_1, \dots, y_l$. Once a norm set is fixed we apply the reduction from Theorem 2. ■

5 Conclusions and Future Research

In this paper we have proposed normative mechanism design as a formal tool for analysing normative environment programs such as 2OPL programs. We have shown how to abstract from a particular environment specification language and how to apply methods from mechanism design to verify whether some norm restrictions imposed on a multi-agent environment agree with the behaviour the designer expects. More precisely, we have introduced normative behaviour functions for representing the "ideal" behaviour of multi-agent environments with respect to different sets of agents' preferences. The latter has enabled us to apply concepts from game theory to identify agents' rational behaviour. These formal ideas can now be used to verify whether a norm set (a set of norms and sanctions) is sufficient to motivate agents to act in such a way that the behaviour described by the normative behaviour function is met.

We have defined a norm set in such a way that is can modify facts in the environment states (cf. Def. 6 and 7). As the language used for modelling agents' preferences and the facts in norm sets are based on the same set of propositional symbols, a normative mechanism can steer the behaviour of each agent in (almost) arbitrary ways. This notion of mechanism is very powerful. A first refinement would be to identify a subset $\Pi_M \subseteq \Pi$ of *facts* and assume that a norm set can only modify state valuations with respect to this set. Such a mechanism can be much weaker but also more natural.

Another direction of future research is to consider *robustness* against *group deviation*. Our approach can be extended such that each agent a has its "own" set Π_a of propositional symbols which is used for its preference formulae. If we now want that some agents are not sensitive to norms and sanctions we simply define the set Π_{NF} of facts that are used in a norm set such that $\Pi_{NF} \cap \Pi_a = \emptyset$. Another alternative is to take on a more game theoretical point of view in the line with [1; 4]. For example, one may consider *partial strategies* which assume that only subgroups of agents play rationally. Then, the outcome is usually not a single path any more, but rather a set of paths. This gives rise to a notion of $(\mathcal{S}, A)$-*implementability*.

We have investigated the problem, given $\mathfrak{M}$ as a $(\mathit{ActSpec}, Q, q^I)$-generated CGS, a set of agents' preferences $\mathcal{P}\mathit{refs}$, and a normative behaviour function f, whether there is a norm set M which $\mathcal{S}$-implements f over $\mathfrak{M}$, q^I and $\mathcal{P}\mathit{refs}$. In future work it would be interesting to find settings in which such norm sets can be constructed efficiently, i.e. "implementing" some norms and sanctions. We also plan to extend our analysis to other implementability notions apart from Nash equilibria, e.g. dominant strategy equilibrium implementability.

Again, we like to emphasize that our work is closely related to [1; 16]. In the former, labelled Kripke structures are considered as models supposing that each agent controls some transitions. A norm is then considered as the deactivation of specific transitions. The main difference to our work is that the outcome is assumed to be independent of the preferences where we consider a more general setting captured in terms of normative behaviour functions. Also, norms in our approach can either deactivate or create a transition. It is also important to recall that our focus is of a more practical nature. We try to implement and to analyse mechanisms from a practical point of view, i.e., how to specify normative multi-agent environments. This is best indicated in the way we construct CGSs: They are generated from action specifications.

References

[1] T. Ågotnes, W. van der Hoek, and M. Wooldridge. Normative system games. In *Proceedings of the AAMAS '07*, pages 1–8, New York, NY, USA, 2007. ACM.

[2] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.

[3] N. Bulling and W. Jamroga. Verifying agents with memory is harder than it seemed. *AI Communications*, 23(4):389–403, December 2010.

[4] N. Bulling, W. Jamroga, and J. Dix. Reasoning about temporal properties of rational play. *Annals of Mathematics and Artificial Intelligence*, 53(1-4):51–114, 2009.

[5] M. Dastani, D. Grossi, J.-J. Ch. Meyer, and N. Tinnemeier. Normative multi-agent programs and their logics. In *Proceedings of KRAMAS 2008*, volume LNAI 5605, pages 16–31. Springer, 2009.

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

[7] M. Esteva, J.A. Rodríguez-Aguilar, B. Rosell, and J.L. Arcos. AMELI: An agent-based middleware for electronic institutions. In *Proceedings of AAMAS 2004*, pages 236–243, New York, US, July 2004.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman: San Francisco, 1979.

[9] G. Gottlob, G. Greco, and F. Scarcello. Pure nash equilibria: hard and easy games. In *Journal of Artificial Intelligence Research*, pages 215–230. ACM Press, 2003.

[10] D. Grossi. *Designing Invisible Handcuffs*. PhD thesis, Utrecht University, SIKS, 2007.

[11] A. J. I. Jones and M. Sergot. On the characterization of law and computer systems. In J.-J. Ch. Meyer and R.J. Wieringa, editors, *Deontic Logic in Computer Science: Normative System Specification*, pages 275–307. John Wiley & Sons, 1993.

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

[13] A. Pnueli. The temporal logic of programs. In *Proceedings of FOCS*, pages 46–57, 1977.

[14] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proceedings AAAI-92*, San Diego, CA, 1992.

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

[16] W. van der Hoek, M. Roberts, and M. Wooldridge. Social laws in alternating time: Effectiveness, feasibility, and synthesis, 2007.