# Formal Frameworks for Verifying Normative Multi-agent Systems

Max Knobbout, Mehdi Dastani, and John-Jules Ch. Meyer(✉)

Department of Information and Computing Sciences, Utrecht University,
P.O. Box: 80.089, 3508 TB Utrecht, The Netherlands
{M.Knobbout,J.J.C.Meyer}@uu.nl

**Abstract.** In this paper we concern ourselves with normative multi-agent systems, which are multi-agent systems governed by a set of norms. In these systems, the internals and architecture of the participating agents may be unknown to us, which disables us to make any strong assumption on the possible behaviour that these agents may exhibit. Thus, we cannot simply assume that the agents are aware of the norms, or that they are compliant with respect to the norms. In other words, a crucial problem that needs to be solved is how we can verify these systems if we have no idea whether the agents will be norm-obedient. This paper investigates two distinct formal frameworks which allow us to tackle this problem, namely in the first part of this paper we propose a logic-based framework which uses compliance types, and in the second part we propose a framework which tackles the problem from a mechanism-design perspective.

## 1 Introduction

A lot of work has contributed to the on-going field of (run-time and offline) verification of programs and systems, such as the verification of object-oriented programs [2] or the verification of agent programming with declarative goals [6]. The field we are interested in are normative multi-agent systems, which are multi-agent systems governed by a set of norms. In the spirit of this work, we are going to explore frameworks for the verification of normative multi-agent systems.

A multi-agent system is a computerized system that is composed of multiple interacting agents within an environment [16]. These systems are generally composed or designed with a specific goal in mind, and depending on the behaviour of the participating agents these goals may, or may not, be achieved. In order to regulate, coordinate and control these systems, norms have been proposed, leading to the field of research called normative multi-agent systems [5]. However, since the internals and architecture of the participating agents may be unknown to us, we cannot simply assume that the agents are aware of the norms, or that they are compliant with respect to the norms. In other words, a crucial problem that needs to be solved is how we can verify these systems if we have no idea whether the agents will be norm-obedient. This paper investigates two distinct

© Springer International Publishing Switzerland 2016
E. Ábrahám et al. (Eds.): de Boer Festschrift, LNCS 9660, pp. 294–308, 2016.
DOI: 10.1007/978-3-319-30734-3_20

formal frameworks which allow us to verify these systems. Formal verification is the act of rigorously proving (or disproving) that the system works as intended. Formal verification is of crucial importance if we are looking for a *guarantee* that the system is correct. Whenever the cost of defection is high, it is of importance that we know that a system is correct without actually having to run it.

In Sect. 2 we briefly introduce the model we use for normative multi-agent systems. In Sect. 3 we consider our first framework that uses compliance types. However, this framework does not take into account *why* the agents would behave norm-compliant, we just *assume* it is the case. In Sect. 4 we consider our second framework that does take these motives into account. Particularly, we assume the agents have some preferences (which are possibly unknown to us), and use solution concepts to predict what the agents will play. This approach tackles the problem from a mechanism design perspective. For a general overview of mechanism design we direct the reader to [13]. In Sect. 5 we discuss the paper.

## 2 Preliminaries

In this section we briefly introduce the model of execution we consider for multi-agent systems. We consider simple transition systems, consisting of states of the world, and a complete labelling of joint-actions over the transitions connecting these states. Moreover, we assume a set of atomic (negative or positive) sanctions, which represent certain fines and rewards we can give to the agents. They will play a role later, when we will introduce the notion of state-based norms.

**Definition 1 (Multi-agent System).** *A multi-agent system $M$ is a tuple $(Q, q_0, S, Ags, act, \delta)$ such that:*

- *$Q$ is a finite set of states.*
- *$q_0 \in Q$ the initial state.*
- *$S$ is a finite set of atomic sanction propositions.*
- *$Ags = \{1, ..., n\}$ is a finite non-empty set of agents.*
- *$act : Ags \times Q \mapsto \mathbb{N}_{>0}$ is a function that assigns to each agent and each state the number of available actions. We identify the actions of agent $i \in Ags$ in state $q \in Q$ with the numbers $1 \dots act(i, q)$. For each state $q \in Q$, a joint action is a vector $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_{|Ags|})$ such that $1 \leq \alpha_i \leq act(i, q)$ for every agent $i \in Ags$. Given a state $q \in Q$, we write $Act(q)$ for the set $\{1, \dots, act(1, q)\} \times \cdots \times \{1, \dots, act(|Ags|, q)\}$ of all possible joint actions.*
- *$\delta$ is a transition function which maps a state $q \in Q$ and joint action $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_{|Ags|}) \in Act(q)$ to the resulting next state $\delta(q, \boldsymbol{\alpha}) \in Q$.*

This model is thus concurrent, synchronous, decentralized, discrete and deterministic, and is similar to the notion of concurrent game structures found in [1]. Note that for the sake of simplicity, we only consider sanction propositions; in a more elaborate model states contain facts of the environment, which are assigned by a valuation function. A state-based norm can be modelled as a function that assigns sanctions to states. Note that a sanction can be a fine (e.g. pay $x$ amount

of money), but it can also be a reward. A state can then be considered 'desired' if the norm merely assigns positive sanctions (rewards) to this state and can be considered 'undesired' if the norm merely assigns negative sanctions (fines) to this state, but in general a norm can assign both positive and negative sanctions. Note This approach is closely related to the approach of [15], who defines 'red' and 'green' states as the desired/undesired states of a system. Formally, we define them as follows.

**Definition 2 (State-Based Norm, Normative Multi-agent System).** *Given a multi-agent system $M = (Q, q_0, S, Ags, act, \delta)$, a state-based norm $\gamma$ is a function $\gamma : Q \mapsto \mathcal{P}(S)$ that maps a state of the multi-agent system to a set of sanction propositions. We write $(M, \gamma)$ for the multi-agent system in which state-based norm $\gamma$ is implemented and refer to such a tuple as a* normative multi-agent system*, and we write $\Gamma_M$ for the set of all possible norms given $M$.*

As usual, a multi-agent system gives rise to a set of possible *runs* (alternatively computations) that can occur. A run, together with a state-based norm, gives rise to an infinite sequence of sanction-sets that occurs along such a run. We call such a sequence an *outcome* of a normative multi-agent system.

**Definition 3 (Runs, Outcomes).** *Given a multi-agent system $M = (Q, q_0, S, Ags, act, \delta)$, a* run *is defined as an infinite sequence $r = q_0 q_1 q_2 \cdots \in Q^\omega$ starting from initial state $q_0$ such that $\forall j \in \mathbb{N}_0$ there exists a joint action $\boldsymbol{\alpha} \in Act(q_j)$ such that $\delta(q_j, \boldsymbol{\alpha}) = q_{j+1}$. The set of all possible runs over $M$ is denoted by $\mathcal{R}_M$. A run $r = q_0 q_1 q_2 \ldots$ and a state-based norm $\gamma$ gives rise to an* outcome *$\gamma(r) = \gamma(q_0)\gamma(q_1)\gamma(q_2)\cdots \in \mathcal{P}(S)^\omega$. We write $\mathcal{O}_M = \{\gamma(r) \mid r \in \mathcal{R}_M \text{ and } \gamma \in \Gamma_M\}$ for the set of all possible outcomes given multi-agent system $M$.*

Thus, a run $r$ and norm $\gamma$ give rise to an outcome $\gamma(r)$. In this system agents can adopt strategies, which are mappings from finite sequences of states to an action of the respective agent. A strategy for each agent, referred to as a strategy profile, gives rise to a unique outcome of the normative multi-agent system.

**Definition 4 (Strategies).** *Given a multi-agent system $M = (Q, q_0, S, Ags, act, \delta)$, a* strategy *for an agent $i \in Ags$ is a mapping $\sigma_i$, mapping a finite sequence of states $q_0, ..., q_k \in Q^+$ to an element of $act(i, q_k)$. A strategy profile $\sigma = (\sigma_1, \ldots, \sigma_{|Ags|})$ is a tuple containing a strategy for each agent. A strategy profile $\sigma$, when executed in $M$, gives rise to a unique run from $\mathcal{R}_M$, and we write $run(\sigma)$ to denote this run.*

Thus, a multi-agent system $M$, a norm $\gamma$ and a strategy profile $\sigma$ give rise to a unique outcome $\gamma(run(\sigma)) \in \mathcal{O}_M$. Using these concepts, in the next section we provide a verification framework that allows us to verify normative multi-agent system using compliance types.

## 3 Verification Framework Using Compliance Types

Traditional offline verification of a multi-agent system typically takes on the following form. We are given a normative multi-agent system $(M, \gamma)$ together

with a set of desired outcomes $\mathcal{O}_{\text{desired}} \subseteq \mathcal{O}_M$, the latter depicting the set of outcomes that are desired by the designer of the system. The objective now is to verify whether the $\mathcal{O}_{\text{desired}}$ is *guaranteed*, i.e., whether it is the case that for all runs $r \in \mathcal{R}_M$ we have that $\gamma(r) \in \mathcal{O}_{\text{desired}}$. Such a set $\mathcal{O}_{\text{desired}}$ usually is specified by some temporal property of the system, for example "$\Box\neg$bad" stating that always nothing bad happens, or "$\Diamond$good" stating that somewhere in the future something good will happen. Linear Temporal Logic (LTL) as proposed in [14] is a logic that allows assertions of this form. An LTL formula $\varphi$ can be evaluated along an outcome $o \in \mathcal{O}_M$ (remember that an outcome is an *infinite* sequence of sanction-sets). We will write $o \models \varphi$ whenever an outcome $o$ satisfies an LTL formula $\varphi$, and assume the reader is familiar with the basics of LTL without explicitly defining the semantics. Verification then asks whether an LTL formula $\varphi$ is valid in a normative multi-agent system $(M, \gamma)$, i.e. whether for all runs $r \in \mathcal{R}_M$ we have that $\gamma(r) \models \varphi$.

Several refinements of LTL have been proposed to extend the possible verification questions one might ask. For example, Computation Tree Logic (CTL), as shown in [8], is a logic that allows explicit (universal and existential) quantification over the set of runs within a logical formula. Later, Alternating-time Temporal Logic (ATL), as introduced in [1], was introduced as an extension of CTL to reason about the possible runs that agents can *enforce*. This language allows even more refined assertions of the form $\langle\langle i \rangle\rangle \varphi$, where $i \in Ags$ is an agent and $\varphi$ is a temporal formula (in actuality the language allows to reason about what *coalitions of agents* can enforce, but we do not need to go into such detail). Such a formula can be read as "agent $i$ can enforce $\varphi$ to be true", and such a formula can be evaluated along a normative multi-agent system. We say that $M, \gamma \models \langle\langle i \rangle\rangle \varphi$ is true if and only if there exists a strategy $\sigma_i$ for agent $i$ such that for all strategies $\sigma_{-i}$ it is the case that $\gamma(run((\sigma_i, \sigma_{-i}))) \models \varphi$. Observe that we use notation $\sigma_{-i}$ to denote the strategies of all the other agents apart from $i$, which together with $\sigma_i$ gives rise to the strategy profile $(\sigma_i, \sigma_{-i})$.

Although these logics allow us to express complex temporal properties, in order to verify normative multi-agent systems an even more refined approach should arguably be taken. In this work, we do not assume that implementing a system of norms enforces every agent to be perfectly norm-obedient. However, a lot of strategy profiles the agents can adopt would be very implausible to occur. For example, in a smart road multi-agent system, it would be very implausible to consider a situation where all the agents would neglect all the norms (i.e. drive on the wrong side of the road). However, it might be plausible to consider that *some* of the agents are neglectful with respect to the norms, while the other agents are obedient. In other words, in order to verify these systems, it is important to consider more refined quantifications over the possible strategies that can occur. For example, is it the case that we are guaranteed that an outcome from $\mathcal{O}_{\text{desired}}$ is reached if all of the agents adopt a norm-obedient strategy? And, is this still the case if one of the agents adopts a strategy which breaks some of the norms? Related to the approach we have taken in [9], in order to express these kinds of properties, we introduce the notion of a *compliance type*.

**Definition 5 (Compliance Types, Compliance Profile).** *Given a multi-agent system $M$ giving rise to a set of possible outcomes $\mathcal{O}_M$, we define a* compliance type *as a function $\tau : \mathcal{O}_M \mapsto \{0, 1\}$ mapping outcomes to either 0 or 1. We say that an outcome $o \in \mathcal{O}_M$ is* $\tau$-compliant *if and only if $\tau(o) = 1$. A compliance profile $\hat{\tau} = (\tau_1, \ldots, \tau_{|Ags|})$ is a tuple containing a compliance type $\tau_i$ for each agent $i$.*

In other words, different notions of compliance can be constructed, and a verifier of the system can choose these freely. A compliance type thus relates state-based norms with compliant behaviour. As an example, suppose we have a sanction atom $v$ denoting some violation in the system. Then, we could define a compliance type $\tau_{\text{"never v"}}$ stating that $v$ should never occur along an outcome as follows:

$$\tau_{\text{"never v"}}(o) = \begin{cases} 1 & \text{if } o \models \Box\neg v \\ 0 & \text{otherwise.} \end{cases}$$

Depending on our verification needs, more elaborate compliance types can also be defined, for example "sometimes $v$", or "at most $n$ times $v$". We can lift the notion of compliant runs to compliant strategies as follows.

**Definition 6 (Compliant Strategies).** *Given a normative multi-agent system $(M, \gamma)$ and compliance type $\tau$, we say that a strategy $\sigma_i$ for agent $i$ is* $\tau$-compliant *if and only if for all strategies $\sigma_{-i}$ it is the case that $\gamma(run((\sigma_i, \sigma_{-i})))$ is $\tau$-compliant.*

Intuitively, a strategy $\sigma_i$ for agent $i$ is $\tau$-compliant if and only if all the outcomes that can occur if agent $i$ would play this strategy are $\tau$-compliant. Since we do not know what the actual compliance behaviour of the agents will be, we verify the system with respect to a set of possible compliance profiles. Using these concepts, we can state a version of the verification problem as follows.

**Verification Problem 1.** *The verification problem asks, given a normative system $(M, \gamma)$ and a set of compliance profiles $T$, whether it is the case that for all $\hat{\tau} = (\tau_i, \tau_{-i}) \in T$ and for all agents $i \in Ags$ there exists a $\tau_i$-compliant strategy $\sigma_i$for agent $i$ such that for all $\tau_{-i}$-compliant strategies $\sigma_{-i}$ it is the case that:*

$$\gamma(run((\sigma_i, \sigma_{-i}))) \in \mathcal{O}_{\text{desired}}$$

In words, this verification problem asks to verify whether for each compliance type, each agent individually has a strategy aligned with this compliance type such that for all strategies of the other agents that are aligned with this compliance type, if the agents would adopt these strategies a desired outcome is reached. We can extend the language of ATL even further to give a logical characterization of the verification questions. Let $\varphi$ be a formula characterizing $\mathcal{O}_{\text{desired}}$ and let $\hat{\tau} = (\tau_i, \tau_{-i})$ be a compliance profile. We say that

$$M, \gamma \models \langle\!\langle i \mid \hat{\tau} \rangle\!\rangle \varphi$$

is valid if and only if there exists a $\tau_i$-compliant strategy $\sigma_i$ for agent $i$ such that for all $\tau_{-i}$-compliant strategies $\sigma_{-i}$ it is the case that $\gamma(run((\sigma_i, \sigma_{-i}))) \models \varphi$

(again, we do not concern ourselves with a formal definition of the underlying semantics). Such a logical language allows us to specify whether certain temporal properties (specified by $\varphi$) are true if the agents would behave according to certain compliant strategies (specified by $\hat{\tau}$). We can then logically characterize our verification task as follows. Given a normative system $(M, \gamma)$, a set of compliance profiles $T$ and a temporal formula $\varphi$ representing the desired outcomes, verify whether:

$$\forall i \in Ags, \forall \hat{\tau} \in T : M, \gamma \models \langle\langle i \mid \hat{\tau} \rangle\rangle \varphi$$

This is related to the approach we take in [9]. In the next section we will look at an example to get some more intuition on how we can use these assertions to solve the verification problem.

### 3.1 Example

We consider the multi-agent system $M$ and norm $\gamma$ depicted in Fig. 1.

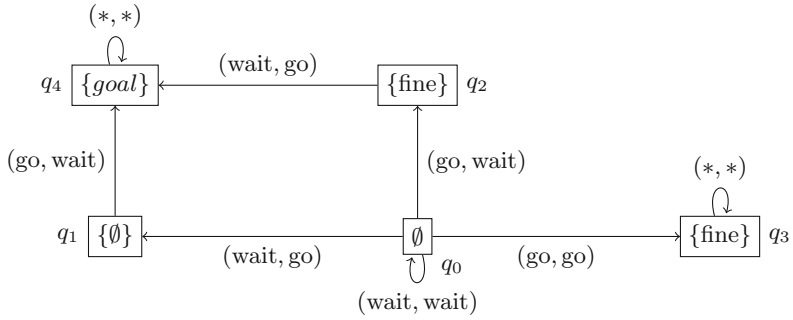


**Fig. 1.** Multi-agent system and norm consisting of two agents each controlling a train at opposite ends of a tunnel. The agents need to coordinate their actions to not perform action 'go' at the same time.

This system consists of two agents who, starting from initial state $q_0$ (middle bottom), can either perform a 'wait' or 'go' action. If they both wait, nothing happens, and if they both go, a dangerous situation occurs. If one of them goes and one of them waits, no dangerous situation occurs. The scenario corresponds to two agents controlling a train at opposite ends of the tunnels, and they must coordinate their actions to not end up in the tunnel at the same time. We have the state-based norm $\gamma$ that assigns a positive sanction *goal* to state $q_4$, i.e. $\gamma(q_4) = \{goal\}$, a negative sanction fine to state $q_2$ and $q_3$, i.e. $\gamma(q_2) = \gamma(q_3) = \{\text{fine}\}$, and does not assign any (positive or negative) sanction to the remaining states. Note that these sanctions in some way reflect that we want the agents to choose the computation $q_0 q_1 (q_4)^\omega$, since this is the only computation for which positive sanction *goal* holds, while containing no negative sanction fine. It is our job to predict whether the agents will indeed, under reasonable assumptions,

choose this computation, using our framework of compliance types. We define the following compliance types, $\tau_{\text{obedient}}$ and $\tau_{\text{neglectful}}$ using temporal logic:

$$\tau_{\text{obedient}}(o) = \begin{cases} 1 & \text{if } o \models \Box\neg\text{fine} \\ 0 & \text{otherwise.} \end{cases}$$

And:

$$\tau_{\text{neglectful}}(o) = 1$$

Thus, every possible outcome $o \in \mathcal{O}_M$ is $\tau_{\text{neglectful}}$-compliant, and this compliance type corresponds to a run in which the agents do not care about the norms. Moreover, given an outcome $o \in \mathcal{O}_M$, we have that $o$ is $\tau_{\text{obedient}}$-compliant if and only if $o \models \Box\neg\text{fine}$. In words, an outcome $o$ is $\tau_{\text{obedient}}$-compliant if it is never the case that sanction fine holds somewhere along $o$. For example, the outcome $\gamma(q_0(q_3)^\omega)$ is not $\tau_{\text{obedient}}$-compliant, because:

$$\gamma(q_0(q_3)^\omega) \not\models \Box\neg\text{fine}$$

As another example, the outcome $\gamma((q_0)^\omega)$ is $\tau_{\text{obedient}}$-compliant, because:

$$\gamma((q_0)^\omega) \models \Box\neg\text{fine}$$

We have for agent 1 that the strategy that always adopts action $w$ at state $q_0$ is $\tau_{\text{obedient}}$-compliant, while all the remaining strategies are not $\tau_{\text{obedient}}$-compliant. To see why this is the case, observe that if agent 1 would adopt *go* at state $q_0$, then agent 2 could for example play action *go* to end up in state $q_3$, at which fine is the case. For agent 2, none of the strategies are $\tau_{\text{obedient}}$-compliant. To see why this is the case, observe that for every strategy $\sigma_2$ agent 2 can select, there exists a strategy $\sigma_1$ for agent 1 such that $\gamma(run((\sigma_1, \sigma_2))) \models \Diamond\text{fine}$. Particularly, whatever agent 2 plays at initial state $q_0$, whenever agent 1 play *go* at this state, we either go to state $q_2$ or state $q_3$, both at which sanction fine is the case.

Now consider the following set $T = \{(\tau_{\text{obedient}}, \tau_{\text{neglectful}})\}$ containing a single compliance profile $(\tau_{\text{obedient}}, \tau_{\text{neglectful}})$. Let $\varphi = \Diamond goal$ be an LTL-formula that characterizes the set of desired runs of the system. Our verification problem now asks whether the following is the case:

$$\forall i \in \{1, 2\} : M, \gamma \models \langle\!\langle i \mid (\tau_{\text{obedient}}, \tau_{\text{neglectful}})\rangle\!\rangle \Diamond goal$$

In order to show whether this is the case, observe that there exists a $\tau_{\text{neglectful}}$-strategy $\sigma_2$ for agent 2, particularly the strategy that plays action *go* at state $q_0$, such that for all $\tau_{\text{obedient}}$-strategies $\sigma_1$ for agent 1, particularly the strategy mentioned earlier that always plays $w$ at state $q_0$, such that $\Diamond goal$ is the case. In formula:

$$M, \gamma \models \langle\!\langle 2 \mid (\tau_{\text{obedient}}, \tau_{\text{neglectful}})\rangle\!\rangle \Diamond goal$$

However, we do not have that there exists a $\tau_{\text{obedient}}$-strategy $\sigma_1$ for agent 1 such for all $\tau_{\text{neglectful}}$-strategies $\sigma_2$ for agent 2 it is the case that $\Diamond goal$. To see this, the only strategy for agent 1 we have to consider is again the one that always plays

$w$ at state $q_0$. However, agent 2 can also play $w$ forever in state $q_0$, resulting in outcome $\gamma((q_0)^\omega)$, for which we have that $\gamma((q_0)^\omega) \not\models \Diamond goal$. Thus, in formula, we have:

$$M, \gamma \not\models \langle\langle 1 \mid (\tau_{\text{obedient}}, \tau_{\text{neglectful}}) \rangle\rangle \Diamond goal$$

In other words, the norm $\gamma$ does not pass our verification test. In particular, this example highlights that the strategic capabilities for the agents may differ given a particular compliance profile.

### 3.2 Framework Discussion

In this section we have given a basic logic-based framework in which normative multi-agent systems can be verified. The logic we briefly described allowed us to assert the verification question, but we have not yet discussed the complexity of such a logic (or the existence of a proof system). However, in [9] we used a similar approach of verifying normative systems by introducing an extension of ATL called an-ATL (abstract normative ATL), but instead of state-based norms we considered transition-based norms, and the compliance types we considered were related to the violation of such norms. We showed that verifying an-ATL formulas remains close to the complexity of ATL, and is thus a suitable candidate logic not only to express, but also to perform the verification task.

## 4 Verification Framework Using Mechanism Design

In the previous section, the verification problem we were concerned with was whether certain (compliant or non-compliant) behaviours of the agents can lead to desired outcomes of the normative multi-agent system. However, this approach does not take into account *why* the agents would behave in such a manner. The agents might have personal preferences that decide how they will behave, and our approach in the previous section does not take these motives into account. Moreover, these preferences might not be known to the designer of the system. Once we make the assumption that the agents have *some* preference, and we are interested in how the agents *will* behave, we enter the field of game theory. Game theory, in the broad sense, is the study of strategic decision making in the presence of (one or more) rational agents, which has its roots in [11]. For an elaborate introduction to the field of game theory, we direct the reader to [13].

Since we have state-based norms that, once implemented, can change the environment of the system, these norms can act as a *mechanism* that can change the underlying game. If we are interested in whether we can design a norm such that the *predicted* outcomes (using game theory) coincides with the *desired* outcomes, we enter the field of mechanism design. In mechanism design, *a mechanism* is, in the general sense, an institution, procedure, protocol or game for generating outcomes. For a general overview of mechanism design we direct the reader to [13], or for an overview that relates mechanism design to computer science, we direct the reader to [12]. In this paper, we consider that the state-based norms are the indirect mechanisms that can change the environment, and

can thus lead to different outcomes. In order to predict the outcomes that will occur, we need to have some notion of what the agents *know* and what the agents *value*, which is referred to as an *agent type*. Note however, that a mechanism designer might not know the true types of the agents. In this paper we are not concerned with knowledge of the agents, i.e., we will just assume the agents have complete and perfect knowledge of the system and participating agents. Thus, an agent type is simply a preference over outcomes of the system, which we define as follows.

**Definition 7 (Preference).** *Given a multi-agent system $M$, a preference of an agent $i \in Ags$, denoted by $\succsim_i$, is a complete reflexive transitive binary relation over outcomes $\mathcal{O}_M$. If for two outcomes $o, o' \in \mathcal{O}_M$ it holds that $o \succsim_i o'$ and $o' \succsim_i o$, we write $o \sim_i o'$, and when $o \succsim_i o'$ and $o' \not\succsim_i o$, we write $o \succ_i o'$. A preference profile $\succsim = (\succsim_1, \ldots, \succsim_{|Ags|})$ consists of a preference of each agent.*

The reading of $o \succsim_i o'$ should be that agent $i$ prefers outcome $o$ at least as much as outcome $o'$. Thus, given a norm $\gamma$ and two runs $r$ and $r'$ of the system, agent $i$ prefers $r$ at least as much as $r'$ whenever $\gamma(r) \succsim_i \gamma(r')$. It is thus already apparent that norms can influence the runs that can occur by making some runs more attractive than other runs for an agent. We can use the preferences to predict the outcomes that will occur, and we can derive these by using concepts from game theory. Whenever we consider a certain type, in order to make a final prediction of the outcomes that will be achieved, we need certain rules that tell us which outcomes will be rationally optimal. In game theory, these formal rules are called the *solution concepts* that can be used for making these predictions. A multitude of these solution concepts exist, but the one we consider in this paper is that of a *Nash Equilibrium*, see e.g. [13].

**Definition 8 (Nash Equilibrium).** *Given a multi-agent system $M$, a norm $\gamma$, a preference profile $\succsim$ and a strategy profile $\sigma = (\sigma_1, \ldots, \sigma_{|Ags|})$, we say that $\sigma$ constitutes a Nash Equilibrium in $M$ if and only if for all $i \in Ags$ and strategies $\sigma_i'$, it holds that $\gamma(run((\sigma_i, \sigma_{-i}))) \succsim_i \gamma(run((\sigma_i', \sigma_{-i})))$. We define $NE_\gamma(\succsim) = \{\gamma(run(\sigma)) \mid \sigma \text{constitutes a Nash Equilibrium in } M\} \subseteq \mathcal{O}_M$ as the set of all NE outcomes given $M$, $\gamma$ and $\succsim$.*

In words, a strategy profile constitutes a Nash Equilibrium if and only if no agent individually can gain something from deviating from their own respective strategy. Again, for a more detailed introduction to this concept, we refer the reader to [13]. The desired outcomes are the outcomes we want to have occur. In the context of normative systems, the desired outcomes are the ones that maintain order in society. For example, a typical criterion one may adopt in an utilitarian society is that wrong-doers should be punished. Because we do not know the true incentives of the agents (remember, the preferences of the agents might be unknown to us), we consider a set of possible preference profiles $\Theta$. A *social choice rule* takes a possible preference profile, and combines these individual preference of the agents to give a set of desired outcomes of the system. In this paper, we assume that such a function is already specified to us, and are

not concerned with whether such a rule can be specified given the criteria that we set on such a function (see for example Arrow's impossibility theorem [3]). This is the domain of social choice theory, which concerns itself with combining individual preference in order to reach social welfare [4].

**Definition 9 (Social Choice Rule).** *Given a multi-agent system $M$ and a set of possible preference profiles $\Theta$, a social choice rule $f : \Theta \mapsto \mathcal{P}(\mathcal{O}_M)$ is a function that maps a preference profile $\succsim \in \Theta$ to a set of outcomes $f(\succsim) \subseteq \mathcal{O}_M$.*

Thus, given a social-choice rule $f$ and preference profile $\succsim$, we say that $f(\succsim)$ are the set of social optimal outcome, which are the outcomes we want to have occur. Because mechanism designers do not know which outcomes are optimal beforehand (the preferences are initially unknown to us), a more cautious approach has to be employed. This information must slowly be generated as the system is executed. The problem here is the fact that the agents in the system may have their own objectives, and may try to behave in a way that hides the truth. A typical goal of a mechanism designer is thus to develop mechanisms that are *incentive compatible*, meaning that the optimal strategy of the participants is to reveal the truth. An example of such a truth-revealing mechanism is Solomon's dilemma, which we will discuss in the next section. Formally, using such a social choice rule, mechanism design defines the following implementability relation.

**Definition 10 (Nash Implementability).** *Given a multi-agent system $M$ and set of possible preference profiles $\Theta$, we say that a norm $\gamma$ NE-implements social choice rule $f$ if and only if for all $\succsim \in \Theta$ it holds that $NE_\gamma(\succsim) \subseteq f(\succsim)$.*

Note that in actuality, the above relation defines that of *weak implementation*. Weak implementation demands that all the predicted outcomes $\mathrm{NE}_\gamma(\succsim)$ are desired, i.e. are in the set $f(\succsim)$. In *full implementation*, we additionally demand that all desired outcomes are predicted, i.e. $\mathrm{NE}_\gamma(\succsim) = f(\succsim)$ for all $\succsim \in \Theta$. However, when considering state-based norms, demanding full implementation might be too strong, since usually a single state-based norm can only generate a small subset of the possible outcomes. The verification problem we consider in this section can now be stated as follows.

**Verification Problem 2.** *The verification problem asks, given a normative system $(M, \gamma)$, a set of possible preference profiles $\Theta$ and a social choice rule $f$, whether it is the case that $\gamma$ NE-implements social choice rule $f$.*

Let us again look at an example to get some more intuition for the various complex notions introduced in this section.

### 4.1 Example

Solomon's dilemma is often used in literature to describe the idea of implementation theory and mechanism design. In this dilemma two women come before him, both claiming to be the mother of a child, and Solomon has to find out who is lying. In this paper we consider the version discussed in [10] where Solomon is

able to give the mothers a fine. Let us first informally explain the problem and its relation to mechanism design. Solomon (the mechanism designer) initially does not know who the real mother is. Based on this he considers two preference profiles, one preference profile that represents the case in which mother 1 would be the real mother, and one preference profile that represents the case in which mother 2 is the real mother (how he constructs these possible preference profiles will be discussed below). But, as we already mentioned, Solomon does not know which of these preference profiles is the true one. Through a state-based norm, he can give the mothers fines in some states, and he can assign the child to one of the mothers in a state. It is clear that if mother 1 is the true mother, then the optimal outcome would be that mother 1 eventually gets the child forever, and that both mothers never receive any fine. If mother 2 is the true mother, then the optimal outcome would be that mother 2 eventually gets the child forever, and again that both mothers never receive any fine. This example makes it clear why a social choice rule is dependent on the preference profile: since we do not know who the real mother is, we cannot simply say that there exists one unique optimal outcome. It is Solomon's job to construct a norm such that the child is eventually given to the true mother forever without any fines given. We assume that Solomon can give a small fine to mother 1 (represented by sanction proposition $\text{fine}_1$) and a big fine to mother 2 (represented by sanction proposition $\text{fine}_2$). Note that $\text{fine}_2$ is tweaked precisely by Solomon such that this sanction is low enough that if mother 2 would be the real mother, she would care more about the child, while if mother 2 would *not* be the real mother, she would care more about the sanction. Of course this requires some accurate and knowledgeable estimations by Solomon, but we assume that he is wise enough to do this. Moreover, since the situation is symmetric, Solomon could have chosen $\text{fine}_1$ and $\text{fine}_2$ the other way around, but this is beyond the point of example. If $\text{child}_i$ represents that the child is given to mother $i$, and if $\succsim$ represents the preference profile in which mother 1 is the true mother and $\succsim'$ the preference profile in which mother 2 is the real mother, then it is Solomon's job to implement the following social choice rule $f$:

$$\begin{aligned} f(\succsim) &= \{o \in \mathcal{O}_M \mid o \models \Diamond\Box(\text{child}_1 \wedge \neg\text{child}_2) \wedge \neg\Diamond(\text{fine}_1) \wedge \neg\Diamond(\text{fine}_2)\} \\ f(\succsim') &= \{o \in \mathcal{O}_M \mid o \models \Diamond\Box(\neg\text{child}_1 \wedge \text{child}_2) \wedge \neg\Diamond(\text{fine}_1) \wedge \neg\Diamond(\text{fine}_2)\} \end{aligned}$$

Solomon assumes that mother 1 always prefers any outcome over any other outcome if in that outcome she receives the child forever. However, if she does not receive the child forever, Solomon assumes that mother 1 prefers any outcome over any other outcome if this outcome does not contain fine $\text{fine}_1$. In order to represent such a preference, we can use the idea presented in [7] of using a preference order over LTL formulas. This preference can then formally be described as follows:

$$(\Diamond\Box\text{child}_1) \succ_1 (\Box\neg\text{fine}_1) \succ_1 \top$$

Such a list gives rise to a preference over outcomes in multi-agent system as follows. Given two arbitrary outcomes $o, o' \in \mathcal{O}_M$, we determine from left to

right the *first* LTL formula that satisfies the outcome. Let us assume that for $o$ this is the formula $\Box\neg\text{fine}_1$ (thus we have that $o \not\models \Diamond\Box\text{child}_1$) and $o'$ this is the formula $\top$ (thus we have that $o' \not\models \Diamond\Box\text{child}_1$ and $o' \not\models \Box\neg\text{fine}_1$). Then, since $(\Box\neg\text{fine}_1) \succ_1 \top$, this would imply that $o \succ_1 o'$. If two outcomes $o$ and $o'$ satisfy the same formula, we say that $o \sim_1 o'$. If the last formula in such a list is $\top$, we know that such a list gives rise to a complete preference over all possible outcomes since this implies that for every possible outcome we can find at least one formula that is satisfied. This particular preference exactly states what we mentioned earlier: mother 1 always prefers any outcome over any other outcome if in that outcome she receives the child forever. However, if this is not the case, mother 1 prefers any outcome over any other outcome if this outcome does not contain fine $\text{fine}_1$.

For mother 2, king Solomon is in doubt about the following two preferences:

$$(\Diamond\Box(\text{child}_2 \wedge \neg\text{fine}_2)) \succ_2 (\Box\neg\text{fine}_2) \succ_2 \top$$
$$(\Diamond\Box\text{child}_2) \succ'_2 (\Box\neg\text{fine}_2) \succ'_2 \top$$

The first preference $\succsim_2$ states that mother 2 prefers an outcome over any other outcome if she is assigned the child without a fine given. If this is not the case, she would rather not receive a fine. Moreover, she does not care about the remaining outcomes. The second preference $\succsim'_2$ states that mother 2 prefers an outcome in which she is assigned the child, regardless of whether this outcome contains a fine or not, while the rest remains the same. In other words, he either considers that mother 2 cares more about the fine than the child, or more about the child than the fine; the first case represents the case in which mother 1 is the real mother, while the second case represents the case in which mother 2 is the real mother. Thus preference profile $\succsim = (\succsim_1, \succsim_2)$ represents the profile in which mother 1 is the real mother, and preference profile $\succsim' = (\succsim_1, \succsim'_2)$ represents the profile in which mother 2 is the real mother.

Now we are ready to give the solution to the problem, which is drawn in Fig. 2. Consider the multi-agent system $M$ and norm $\gamma$ depicted here. This system consists of two agents who, starting from initial state $q_0$ (below left), can claim to

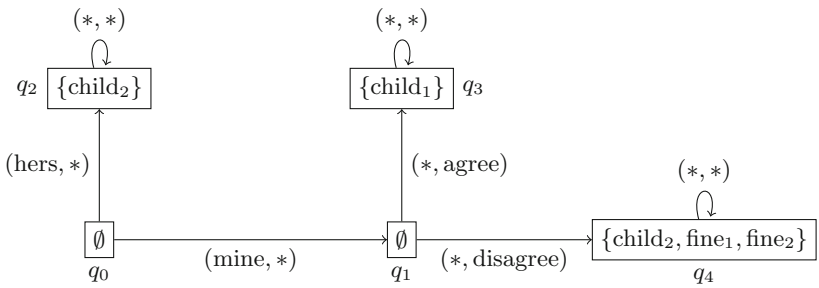


**Fig. 2.** Multi-agent system and norm consisting of two agents claiming whether they are the real mother.

be the real mother or not. Mother 1 can claim in $q_0$ that the child either belongs to her (action 'mine'), or that it belongs to the other mother (action 'hers'). If she performs action 'hers', mother 2 can either agree with this (action 'agree') or disagree with this (action 'disagree'). As can be seen in the figure, Solomon has implemented the state-based norm $\gamma$ that assigns the child to mother 2 in $q_2$, i.e. $\gamma(q_2) = \{\text{child}_2\}$, assigns the child to mother 1 in $q_3$, i.e. $\gamma(q_3) = \{\text{child}_1\}$, and assigns the child to mother 2 while giving a small fine to mother 1 (sanction $\text{fine}_1$) and a big fine to mother 2 (sanction $\text{fine}_2$) in state $q_4$, i.e. $\gamma(q_4) = \{\text{child}_2, \text{fine}_1, \text{fine}_2\}$. But, is it indeed true that for $\Theta = \{\succsim, \succsim'\}$ we have that $\gamma$ NE-implements social choice rule $f$? We will show that this is indeed the case in the remainder of this section.

Observe that in the normative multi-agent system, there exists only two possible strategies for each agent, which we refer to as $\sigma_{\text{hers}}$ and $\sigma_{\text{mine}}$ for mother 1, and $\sigma_{\text{agree}}$ and $\sigma_{\text{disagree}}$ for mother 2. The corresponding outcomes are the following:

| $\gamma$ | $\sigma_{\text{agree}}$ | $\sigma_{\text{disagree}}$ |
|---|---|---|
| $\sigma_{\text{hers}}$ | $\gamma(q_0(q_2)^\omega)$ | $\gamma(q_0(q_2)^\omega)$ |
| $\sigma_{\text{mine}}$ | $\gamma(q_0q_1(q_3)^\omega)$ | $\gamma(q_0q_1(q_4)^\omega)$ |

Consider preference profile $(\succsim_1, \succsim_2)$. We have that $(\sigma_{\text{mine}}, \sigma_{\text{agree}})$ constitutes a Nash Equilibrium. To see why, observe that $\gamma(run((\sigma_{\text{mine}}, \sigma_{\text{agree}}))) \succ_1 \gamma(run((\sigma_{\text{hers}}, \sigma_{\text{agree}})))$ because:

$$\gamma(run((\sigma_{\text{mine}}, \sigma_{\text{agree}}))) \models \Diamond\Box\text{child}_1\text{,and,}$$
$$\gamma(run((\sigma_{\text{hers}}, \sigma_{\text{agree}}))) \models \neg\Diamond\Box\text{child}_1$$

Moreover, observe that $\gamma(run((\sigma_{\text{mine}}, \sigma_{\text{agree}}))) \succ_2 \gamma(run((\sigma_{\text{mine}}, \sigma_{\text{disagree}})))$ because:

$$\gamma(run((\sigma_{\text{mine}}, \sigma_{\text{agree}}))) \models \neg\Diamond\Box(\text{child}_2 \wedge \neg\text{fine}_2) \wedge \Box\neg\text{fine}_2\text{,and,}$$
$$\gamma(run((\sigma_{\text{mine}}, \sigma_{\text{disagree}}))) \models \neg\Diamond\Box(\text{child}_2 \wedge \neg\text{fine}_2) \wedge \neg\Box\neg\text{fine}_2$$

It is also not hard to verify that $(\sigma_{\text{mine}}, \sigma_{\text{agree}})$ is the only NE strategy profile, implying that $\text{NE}_\gamma((\succsim_1, \succsim_2)) = \{\gamma(run((\sigma_{\text{mine}}, \sigma_{\text{agree}})))\}$. Now consider strategy profile $(\succsim_1, \succsim_2')$. We have that $(\sigma_{\text{hers}}, \sigma_{\text{disagree}})$ constitutes a Nash Equilibrium. To see why, observe that $\gamma(run((\sigma_{\text{hers}}, \sigma_{\text{disagree}}))) \succ_1 \gamma(run((\sigma_{\text{mine}}, \sigma_{\text{disagree}})))$ because:

$$\gamma(run((\sigma_{\text{hers}}, \sigma_{\text{disagree}}))) \models \neg\Diamond\Box\text{child}_1 \wedge \Box\neg\text{fine}_1\text{,and,}$$
$$\gamma(run((\sigma_{\text{mine}}, \sigma_{\text{disagree}}))) \models \neg\Diamond\Box\text{child}_1 \wedge \neg\Box\neg\text{fine}_1$$

Moreover, observe that $\gamma(run((\sigma_{\text{hers}}, \sigma_{\text{disagree}}))) \sim_2' \gamma(run((\sigma_{\text{hers}}, \sigma_{\text{agree}})))$ because:

$$\gamma(run((\sigma_{\text{hers}}, \sigma_{\text{disagree}}))) \models \Diamond\Box(\text{child}_2 \wedge \neg\text{fine}_2)\text{,and,}$$
$$\gamma(run((\sigma_{\text{hers}}, \sigma_{\text{agree}}))) \models \Diamond\Box(\text{child}_2 \wedge \neg\text{fine}_2)$$

It is again also not hard to verify that $(\sigma_{\text{hers}}, \sigma_{\text{disagree}})$ is the only NE strategy profile, implying that $\text{NE}_\gamma((\succsim_1, \succsim_2')) = \{\gamma(run((\sigma_{\text{hers}}, \sigma_{\text{disagree}})))\}$. Since we have that:

$$\gamma(run((\sigma_{\text{mine}}, \sigma_{\text{agree}}))) \models \Diamond\Box(\text{child}_1 \wedge \neg\text{child}_2) \wedge \neg\Diamond(\text{fine}_1) \wedge \neg\Diamond(\text{fine}_2), \text{and,}$$
$$\gamma(run((\sigma_{\text{hers}}, \sigma_{\text{disagree}}))) \models \Diamond\Box(\neg\text{child}_1 \wedge \text{child}_2) \wedge \neg\Diamond(\text{fine}_1) \wedge \neg\Diamond(\text{fine}_2)$$

We can conclude that $\text{NE}_\gamma(\succsim) \subseteq f(\succsim')$ and $\text{NE}_\gamma(\succsim') \subseteq f(\succsim')$ as needed. In other words, we have verified that it is indeed the case that, given possible preference profiles $\Theta = \{\succsim, \succsim'\}$,we have that $\gamma$ NE-implements social choice rule $f$: the normative multi-agent system ensures that the child is eventually given to the real mother forever without any fines given.

### 4.2 Framework Discussion

In this section, we discussed how we can frame the verification problem of an multi-agent system using concepts from mechanism design. This idea is related to the work in [7], in which they call this "normative mechanism design". We believe that this field of research is an exciting new way in which normative systems can be studied: norms are viewed as a mechanism constituting a game, allowing us to state the verification problems (in the context of formal verification) as implementation problems (in the context of mechanism design).

## 5 Discussion

In this paper we have presented two distinct verification frameworks in which the correctness of normative multi-agent systems can be (dis)proven. In the first part we used compliance types, and showed how properties of a system can be expressed (and to marginal extent proven) using these types. In the second part we used mechanism design, and showed how implementability questions can be expressed (and to a marginal extend proven). Although both these approaches use norms as a mechanism to steer agents away (or towards) certain outcomes of the system, the main difference is the following:

- With compliance types, we assume certain compliance behaviour of the agents with respect to the norm. We do not know what the actual compliance behaviour of the agents will be, so we verify the system with respect to a set of possible compliance profiles.
- With mechanism design, we assume a certain preference relation over outcomes of the system. We do not know what the true preference of the agents are, so we verify the system with respect to a social choice rule, a solution concept and set of possible preference profiles.

These different approaches offer a generic starting point for which the verification task of normative multi-agent systems can be tackled. As we already mentioned in the introduction, normative systems are making their way into our

everyday life. Formal verification is of crucial importance if we are looking for a *guarantee* that the system is correct. Whenever the cost of defection is high, it is of importance that we know that a system is correct without actually having to run it. Verification gives us this guarantee. Development of such methods and tools play an important role in the advancement of normative multi-agent systems and Artificial Intelligence in general.

## References


1. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**(5), 672–713 (2002)
2. Apt, K.R., de Boer, F.S., Olderog, E., de Gouw, S.: Verification of object-oriented programs: a transformational approach. J. Comput. Syst. Sci. **78**(3), 823–852 (2012)
3. Arrow, K.J.: A difficulty in the concept of social welfare. J. Polit. Econ. **58**(4), 328–346 (1950)
4. Arrow, K.J.: Social Choice and Individual Values. Yale University Press, New Haven (1951)
5. Boella, G., van der Torre, L., Verhagen, H.: Introduction to normative multiagent systems. Comput. Math. Organ. Theor. **12**(2–3), 71–79 (2006)
6. de Boer, F.S., Hindriks, K.V., van der Hoek, W., Meyer, J.J.C.: A verification framework for agent programming with declarative goals. J. Appl. Logic **5**(2), 277–302 (2007)
7. Bulling, N., Dastani, M.: Verifying normative behaviour via normative mechanism design. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 2011), pp. 103–108 (2011)
8. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching-time temporal logic. In: Logic of Programs Workshop, pp. 52–71 (1982)
9. Knobbout, M., Dastani, M.: Reasoning under compliance assumptions in normative multiagent systems. In: Proceedings of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), pp. 331–340 (2012)
10. Moore, J.: Implementation, contracts and renegotiation in environments with complete information. Adv. Econ. Theor **1**, 182–282 (1992)
11. Neumann, J.V., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, Princeton (1944)
12. Nisan, N.: Introduction to mechanism design (for computer scientists). In: Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. (eds.) Algorithmic Game Theory, pp. 209–242. Cambridge University Press, New York (2007)
13. Osborne, M., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge (1994)
14. Pnueli, A.: The temporal logic of programs. In: Proceedings of the 18th Annual Symposium on Foundations of Computer Science, pp. 46–57 (1977)
15. Sergot, M.J.: Action and agency in norm-governed multi-agent systems. In: Artikis, A., O'Hare, G.M.P., Stathis, K., Vouros, G. (eds.) ESAW 2007. LNCS (LNAI), vol. 4995, pp. 1–54. Springer, Heidelberg (2008)
16. Wooldridge, M.: An Introduction to MultiAgent Systems, 2nd edn. Wiley Publishing, Chichester (2009)