Artificial Intelligence 239 (2016) 97–142

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Norm-based mechanism design

Nils Bulling [a,*], Mehdi Dastani [b]

[a] Department of Intelligent Systems, Delft University of Technology, Delft, The Netherlands
[b] Intelligent Systems Group, Utrecht University, Utrecht, The Netherlands

ARTICLE INFO

Article history:
Received 21 September 2015
Received in revised form 4 July 2016
Accepted 7 July 2016
Available online 14 July 2016

Keywords:
Norms
Multi-agent systems
Mechanism design

ABSTRACT

The increasing presence of autonomous (software) systems in open environments in general, and the complex interactions taking place among them in particular, require flexible control and coordination mechanisms to guarantee desirable overall system level properties without limiting the autonomy of the involved systems. In artificial intelligence, and in particular in the multi-agent systems research field, social laws, norms, and sanctions have been widely proposed as flexible means for coordinating the behaviour of autonomous agents in multi-agent settings. Recently, many languages have been proposed to specify and implement norm-based environments where the behaviour of autonomous agents is monitored, evaluated based on norms, and possibly sanctioned if norms are violated.

In this paper, we first introduce a formal setting of multi-agent environments based on concurrent game structures 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 a specific behaviour can be enforced (or implemented) if agents follow their subjective preferences. We relate concepts from mechanism design to our setting, where agents' preferences are modelled by linear time temporal logic (LTL) formulae. This proposal bridges the gap between norms and mechanism design allowing us to formally study and analyse the effect of norms and sanctions on the behaviour of rational agents. The proposed machinery can be used to check whether specific norms and sanctions have the designer's expected effect on the rational agents' behaviour or if a set of norms and sanctions that realise the effect exists at all. We investigate the computational complexity of our framework, focusing on its implementation in Nash equilibria and we show that it is located at the second and third level of the polynomial hierarchy. Despite this high complexity, on the positive side, these results are in line with existing complexity results of related problems. Finally, we propose a concrete executable specification language that can be used to implement multi-agent environments. We show that the proposed specification language generates specific concurrent game structures and that the abstract multi-agent environment setting can be applied to study and analyse the behaviour of multi-agent programs with and without norms.

* Corresponding author.
E-mail addresses: n.bulling@tudelft.nl (N. Bulling), M.M.Dastani@uu.nl (M. Dastani).

http://dx.doi.org/10.1016/j.artint.2016.07.001
0004-3702/

1. Introduction

The emergence of autonomous software systems and their increasing number of interactions with open environments such as the Internet, financial markets, large ICT systems, socio-technical systems and industrial platforms, urgently requires flexible control and coordination mechanisms in order to guarantee desirable overall system level properties. This urgency became painfully clear in 2010 by the so-called "Flash Crash", where the uncontrolled and uncoordinated interactions between high frequency algorithmic trading systems in financial environments have led to extraordinary upheaval of U.S. equity markets [63]. Similar urgency is experienced in large ICT systems of organisations such as banks or insurance companies, where due to competition and innovation business processes have to be modified in rapid tempo. Such continuous changes constitute a potential threat for business processes to become non-compliant with the companies' business rules and policies. There is, therefore, an increasing need for flexible control and supervision mechanisms that could ensure the compliance of business processes in such dynamic environments without limiting the functionality and performance of the business processes [47,25,49]. In addition to these existing cases, the rapid development of autonomous cars strongly suggests that future traffic will be populated by intelligent autonomous cars that interact in shared (physical) environments called smart roads. In order to guarantee the safety and throughput of such smart road environments, the behaviour of autonomous cars has to be regulated by intelligent monitoring and coordination mechanisms without directly steering their behaviour [40,38]. These and other applications show the urgency of tools and techniques to design, develop and analyse intelligent flexible control and coordination mechanisms.

These and many other applications can be best modelled as multi-agent systems. A multi-agent system consists of interacting computer systems that are situated in some environment and are capable of autonomous actions in the environment in order to meet their delegated objectives [68]. The individual agents are generally assumed to be heterogeneous in the sense that they may be designed and developed by various parties, using different technologies, and pursuing different objectives. It is exactly the autonomous and heterogeneous character of the computer agents in multi-agent applications that require intelligent flexible control and coordination mechanisms. In general, any control and coordination mechanism used in a multi-agent application should strike a balance between the autonomy of the agents on the one hand and the desirable global properties of the multi-agent system on the other hand, i.e., while the autonomy of agents should be respected, the global properties of multi-agent systems should be ensured.

Existing coordination techniques in computer science, such as synchronization techniques or interaction protocols, can be used to ensure the overall desirable properties of the interacting systems. However, these techniques will severely limit the autonomy and intelligence of the involved systems [29]. In artificial intelligence, norms and norm enforcement have been widely proposed as flexible and effective means for coordinating the behaviour of autonomous agents in multi-agent systems [50,45,14]. Singh et al. [62] provide an overview of various uses and applications of norms in multi-agent systems, and Criado et al. [26] discuss some challenges and open issues concerning representation, reasoning, creation and implementation of norms in multi-agent systems. A multi-agent system that uses norms to coordinate the behaviour of agents is often called norm-based multi-agent system or normative multi-agent system.

In general, there are two approaches to exploit norms for coordination purposes in multi-agent systems. Norms can be either endogenous to agents in the sense that they form an integral part of the agents' specifications [59,60], or exogenous to agents in the sense that they are enforced by some external regulatory mechanism [2,27]. Each approach comes with specific assumptions and applications. For example, the endogenous approach assumes that norms are internalised by the agents in the sense that the agents' decision making mechanisms are designed and developed based on a given set of norms. This assumption implies that norms are available at design time and enforced on agents by the agents' developers at design time. The endogenous approach can, for example, be used to examine the (emergent) effects of norms in agent-based simulations [59,60,48,6]. In contrast, the exogenous approach is agnostic about norm internalisation, but assumes an authority that monitors agents' behaviour and enforces norms by means of preventing norm violations or imposing sanctions on violating behaviour [65,2,44,57]. Following the exogenous approach, autonomous systems may respect norms or decide to violate them in which case they may incur sanctions. It is exactly the incursion of sanctions that may incentivize, but not restrict, agents to behave in a particular way. The exogenous approach is also conceived as a way to make the engineering of multi-agent systems easier to manage as it supports the general principle of 'separation of concerns', i.e., it supports the encapsulation of coordination and control concerns in a system entity that is separable from the agents' internals [10,17,70]. In addition, the external authority can be conceived as a mechanism that is designed to implement norms and the norm enforcement process. This perspective on norms and norm enforcement allows us to apply formal tools from game theory and to study and analyse norms and norm enforcement from a mechanism design perspective [18,19,30,69].

This paper follows the exogenous approach and provides a mechanism design perspective on norms and norm enforcement. A multi-agent environment is modelled as a concurrent game structure where possible paths in the game structure denote possible execution traces of the corresponding multi-agent system. By considering the states of the game structure as the states of the environment in which agents operate, the concurrent game structure becomes the representation of a mechanism (game form) that specifies the effect of the agents' actions on the multi-agent environment. The enforcement of norms on a multi-agent system may change the effect of the agents' actions on the environment and thereby its underlying concurrent game structure and the set of possible execution traces. We call the modification of concurrent game structure by norm enforcement *norm-based update*. Clearly, various sets of norms can be used to update a concurrent game structure. The decision regarding which set of norms to use depends on the behaviour of the agents which in turn is a

result of the agents' preferences, and, of course, the intended objective of the multi-agent system. We introduce *norm-based mechanism design* as a formal methodology to analyse norms and norm-based updates, and to find suitable norms taking into consideration agents' rational behaviour.

We aim at bridging the gap between norms and mechanism design by focusing on the relation between multi-agent systems, norm enforcement, and concurrent game structures. This relation sets the stage for studying formal properties of norm enforcement such as whether enforcing a set of norms implements a specific social choice function, which describes the desired system executions in relation to the agents' preferences, in specific equilibria. This also allows us, for example, to analyse whether a group of agents is willing to obey some norms. The formal analysis is closely related to work presented in [1] and [65], where the enforcement of norms is modelled by the deactivation of transitions. Adding norms to environments in our model may, however, either deactivate transitions in the case of regimenting norms or change the effect of actions in the case of sanctioning norms. The latter is achieved by (re)labelling states with new propositions that encode, for example, sanctions or incentives which can affect the behaviour of agents. Our work is also motivated by [59] and [60], where social laws were proposed to be used in computer science to control agents' behaviour.

The proposed norm-based mechanism design is an abstract methodology as it assumes concurrent game structures as models of multi-agent environments. In order to ground this methodology, we design an executable specification language to support the development of norm-based multi-agent systems. The aim of the executable specification language is to facilitate the implementation of multi-agent environments that are governed by norms. Multi-agent environments without norms are often implemented using a computational specification language that provides constructs to specify (initial) environment states and actions. The execution of such an environment specification is a process that continuously observes agents' actions and updates the environment state based on the specification of the observed actions. The implementation of multi-agent environments with norms requires additional constructs to specify norms and sanctions. The execution of an environment specification with norms includes two additional steps through which the observed agents' actions are further evaluated based on the given set of specified norms after which norm violations are either prevented or sanctioned. Our abstract norm-based mechanism design methodology can be applied to norm-based multi-agent environment programs in order to analyse whether the enforcement of a set of norms by a norm-based environment program can implement a specific social choice function. We also investigate the complexity of verifying whether a given norm-based multi-agent system implements a given system specification. Besides the verification problem we analyse the decision problem whether a norm-based multi-agent system with desirable properties exists at all. In terms of complexity our results are negative, in the sense that the problems are intractable. On the positive side however, the results are in line with existing complexity results of related problems.

The novel contribution of this work is a formal methodology for analysing norms and norm enforcement used for coordinating the behaviour of rational agents. This is realised by applying game theory/mechanism design techniques to study the effects of logic-based norms on multi-agent systems. We ground the theoretical analysis in an executable multi-agent setting to allow the development of norm-based coordination mechanisms for multi-agent systems. This paper extends and revises the work presented in [19]. The idea of norm-based mechanism design was first proposed in the extended abstract [18]. In comparison with [19] we significantly revise the formal setting, add new complexity results to new decision problems (weak and strong implementability) and give full proofs. In addition, the executable specification language as well as the related work section and a running example are new. The design of the executable specification language is inspired by the programming languages proposed in [28] and [27] for which an interpreter, called 2OPL (Organisation-Oriented Programming Language), has been developed.[1] One of the main differences between these languages and our proposed executable specification language is the representation of norms. While norms in [28] and [27] are state-based (i.e., norms represent prohibited states), the proposed specification language in this paper considers conditional action-based norms (i.e., norms represent prohibited actions in specific states).

The structure of this paper is as follows. First, Section 2 presents concurrent game structures as models for multi-agent environments and connects it to game theory. Section 3 introduces a specification language for norm-based multi-agent environments, extends the formal setting of environments with norms and sanctions, and introduces the concept of *norm-based mechanism design*. Section 4 investigates the complexity of two implementation problems. Section 5 grounds the proposed approach by linking it to a norm-based multi-agent environment programming language. Finally, related work is discussed and some conclusions are drawn. The formal proofs about the computational complexity can be found in the appendix.

2. Multi-agent environment model

In order to illustrate our proposal, we shall use the following example throughout the paper. The example is closely related to the well-known train-gate controller example presented by [9]. Though, we slightly modify it to highlight the rational, decentralized decision making aspect of the car drivers.

Example 1 *(Narrow road example).* The scenario consists of a narrowed road and two cars at the opposite ends of the road. The cars cannot simultaneously pass through the narrowed road. In this scenario, both cars can wait for each other forever,

[1] http://oopluu.sourceforge.net/.

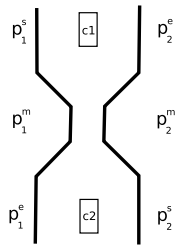

Fig. 1. Road scenario.

both can move simultaneously causing a congestion/crash at the middle of the road, or one can move while the other is waiting. There are two alternatives for the latter case: either the first car moves while the second is waiting or vice versa. The system designer prefers one of these two alternatives. In order to realise this behaviour, the system designer aims at synthesizing a set of norms and corresponding sanctions such that when they are enforced the behaviour of the cars satisfies the given (system) preference assuming that the drivers' behaviours are driven by their own individual preferences. This scenario is illustrated in Fig. 1, where proposition p_i^x indicates that car i is at position $x \in \{s, m, e\}$ (s: start position, m: middle position, e: end position). We assume the following states in this scenario:

- q_0: the cars are at their starting positions, i.e., $p_1^s \wedge p_2^s$ holds
- q_1: car 1 is at its ending and car 2 is at its starting position, i.e., $p_1^e \wedge p_2^s$ holds
- q_2: car 1 is at its starting and car 2 is at its ending position, i.e., $p_1^s \wedge p_2^e$ holds
- q_3: the cars are congested in the middle of the road, i.e., $p_1^m \wedge p_2^m$ holds
- q_4: the cars are at their ending positions, i.e., $p_1^e \wedge p_2^e$ holds

2.1. Concurrent structures and strategies

In the following we introduce *concurrent game structures (*CGSs*)* from [9] (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, possibly including an initial state, and a set of (synchronized and concurrent) transitions. Informally speaking, a CGS is given by a labelled transition system where transitions are activated by action profiles.

Definition 1 *(*CGS*, pointed).* A *concurrent game structure* (CGS) is a tuple $\mathfrak{M} = (\mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o)$, comprising a nonempty finite set of all agents $\mathbb{A}\mathrm{gt} = \{1, \ldots, k\}$, a nonempty finite set of states Q, a nonempty finite set of atomic propositions Π (also called propositional symbols) and their valuation $\pi : Q \to \mathcal{P}(\Pi)$, and a nonempty finite set of (atomic) actions Act. Function $d : \mathbb{A}\mathrm{gt} \times Q \to \mathcal{P}(Act)\backslash\{\emptyset\}$ defines nonempty sets of actions available to agents at each state, and o is a (deterministic) transition function that assigns the outcome state $q' = o(q, (\alpha_1, \ldots, \alpha_k))$ to state q and a tuple of actions $(\alpha_1, \ldots, \alpha_k)$ with $\alpha_i \in d(i, q)$ and $1 \leq i \leq k$, that can be executed by $\mathbb{A}\mathrm{gt}$ in q. A *pointed CGS* is given by $(\mathfrak{M}, q)$ where $\mathfrak{M}$ is a CGS and q is a state in it.

In the following, we write $d_i(q)$ instead of $d(i, q)$ and $o(q, \vec{\alpha})$ instead of $o(q, (\alpha_1, \ldots, \alpha_k))$ for $\vec{\alpha} = (\alpha_1, \ldots, \alpha_k)$. In CGSs it is assumed that all the agents execute their actions synchronously.[2] The combination of actions together with the current state determines the next transition of the system.

Example 2 *(*CGS*).* Our scenario from Example 1 is formally modelled by CGS $\mathfrak{M}_1 = (\mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o)$, shown in Fig. 2, where

- $\mathbb{A}\mathrm{gt} = \{1, 2\}$,
- $Q = \{q_0, \ldots, q_4\}$,
- $\Pi = \{p_i^x \mid i \in \{1, 2\} \text{ and } x \in \{s, m, e\}\}$,
- $Act = \{M, W\}$,
- the function d is defined as

[2] We note that the framework allows to model turn-based games as a special case.

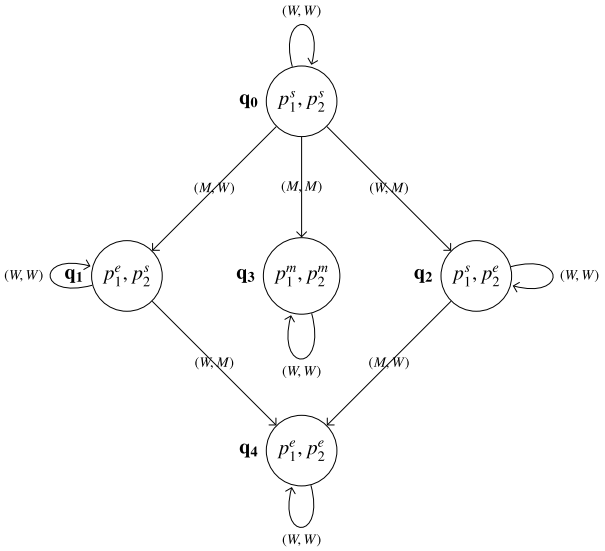

Fig. 2. The CGS $\mathfrak{M}_1$ modelling the scenario from Example 1. Nodes represent system states, bold symbols $\mathbf{q_i}$ assigned to nodes are state names, and elements p_i^x within a node are atomic propositions that hold in the state.

$$\begin{array}{ll} d_1(q_0) = \{W, M\} & d_2(q_0) = \{W, M\} \\ d_1(q_1) = \{W\} & d_2(q_1) = \{W, M\} \\ d_1(q_2) = \{W, M\} & d_2(q_2) = \{W\} \\ d_1(q_3) = \{W\} & d_2(q_3) = \{W\} \\ d_1(q_4) = \{W\} & d_2(q_4) = \{W\} \end{array}$$

- π and o are defined as illustrated in Fig. 2, e.g. $o(q_0, (M, M)) = q_3$.

We note that agent 1 and 2 have no choice other than to wait (W) in states $\{q_1, q_3, q_4\}$ and states $\{q_2, q_3, q_4\}$, respectively.

In the following we assume that a CGS $\mathfrak{M} = (\mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o)$ is given, if not said otherwise. A *strategy* of agent i is a conditional plan that specifies what i is going to do in each situation. It makes sense, from a conceptual and computational point of view, to distinguish between two types of "situations" (and hence strategies): an agent might base its decision only on the current state or on the whole history of events that have happened. In this paper we only consider the former type of strategies which are often called memoryless or positional. In general, memoryless strategies enjoy better computational properties than the second type of strategies called perfect-recall strategies. Note that 'memoryless' sounds more severe than it actually is: an agent is still able to base decisions on the current state; thus, finite histories up to an arbitrary but fixed length could be encoded within states.

Definition 2 *(Strategy).* A (memoryless) *strategy* for agent i is a function $s_i : Q \to Act$ such that $s_i(q) \in d_i(q)$. The set of such strategies is denoted by Σ_i. A *collective strategy* for a group of agents $A = \{i_1, \ldots, i_r\} \subseteq \mathbb{A}\mathrm{gt}$ is a tuple[3] $s_A = (s_{i_1}, \ldots, s_{i_r})$ where each s_{i_j}, $1 \leq j \leq r$, is a strategy for agent i_j. The set of A's collective strategies is given by $\Sigma_A = \prod_{i \in A} \Sigma_i$. The set of all *(complete) strategy profiles* is defined as $\Sigma = \Sigma_{\mathbb{A}\mathrm{gt}}$.

Given the notation above, (s_1, s_2) is a collective strategy of agents 1 and 2. A *path* $\lambda = q_0 q_1 \ldots \in Q^\omega$ is an infinite sequence of states such that for each $j = 0, 1, \ldots$ there is an action tuple $\vec{\alpha} \in d_{\mathbb{A}\mathrm{gt}}(q_j)$ with $o(q_j, \vec{\alpha}) = q_{j+1}$. The set of all paths starting in state q is denoted by $\Lambda_{\mathfrak{M}}(q)$. We write $\lambda[j]$ to refer to the j-th state on path λ, and $\lambda[j, \infty]$ to refer to the (sub)path $q_j q_{j+1} \ldots$ of λ. Function $out_{\mathfrak{M}}(q, s)$ returns the unique path that occurs when the agents execute (the complete) strategy profile s from state q onward.

Definition 3 *(Outcome).* The *outcome* $out_{\mathfrak{M}}(q, s)$ *of a complete strategy profile* $s = (s_1, \ldots, s_k)$ *from state* q *in model* $\mathfrak{M}$ is the path $\lambda = q_0 q_1 q_2 \ldots$ such that $q_0 = q$ and for each $j = 0, 1, 2, \ldots$ there is an action tuple $\vec{\alpha}^j = (\alpha_1^j, \ldots, \alpha_k^j)$ with $\alpha_i^j = s_i(q_j)$ for every $i \in \mathbb{A}\mathrm{gt}$, and $o(q_j, \vec{\alpha}^j) = q_{j+1}$. Often, we will omit subscript "$\mathfrak{M}$" from $out_{\mathfrak{M}}(q, s)$ if clear from context.

The following example illustrates how strategies in our example scenario can be represented.

[3] We assume some implicit ordering among the agents to obtain a unique representative of a strategy profile.

Example 3 *(Strategies).* In the CGS from Example 2, we encode individual strategies by a 5-tuple prescribing an action for each state. However, as agent 1 (resp. agent 2) has no choice other than to wait (W) in states q_1, q_3, and q_4 (resp. q_2, q_3, and q_4), we encode strategies by a tuple xy, indicating that agent 1 (resp. agent 2) executes actions x in q_0 and y in q_2 (resp. x in q_0 and y in q_1). The strategy MW for agent 1 selects M in state q_0 and W in state q_2 (and also W for states q_1, q_3, and q_4). Each agent has therefore 4 strategies resulting in a total of 16 different strategy profiles. Note that a strategy defines an action for all possible states even those not reachable if the current strategy is implemented. Therefore, some strategy profiles have identical outcome paths. For example, all strategies profiles in the set $\{(M\star_1, M\star_2) \mid \star_1, \star_2 \in \{W, M\}\}$ result in the path $q_0 q_3^\omega$.

2.2. Agents' preferences

In the behavioural analysis of multi-agent systems, preferences of agents are often of utmost importance. They are the driving force of agents' behaviour. In the models we are considering, the agents' preferences are defined on the executions of the environment and thus paths in the corresponding CGS. Hence, we assume that agents prefer some executions over others. Therefore, we use temporal formulae to describe sets of paths. This idea was already followed in several pieces of work for similar purposes, e.g. [1] used **CTL** to represent agents' preferences and [23,21] used **ATL** for the same purpose, where [65] used **ATL** for the representation of the objective of the social law. In this paper we use linear temporal logic **LTL**, first proposed by [56] for the verification of programs, for modelling preferences of agents. The logic extends propositional logic with operators that allow to express temporal patterns over infinite sequences of sets of propositions. It allows to express natural properties related to safety, liveness and fairness properties and combinations thereof. As we aim at evaluating system paths, which are infinite sequences of states, we need a logic which allows to compare paths rather than a logic the formulae of which are evaluated in states. This makes **LTL** a more natural choice than, e.g., **CTL** and **ATL**. The basic temporal operators are $\mathcal{U}$ (*until*), $\Box$ (*always*), $\Diamond$ (*eventually*) and $\bigcirc$ (*in the next state*). As before, propositions, drawn from a finite and non-empty set Π, are used to describe properties of states.

Definition 4 *(Language **LTL**).* Let Π be a finite, non-empty set of propositions. The *formulae of* $\mathbf{LTL}_\Pi$ are generated by the following grammar, where $\mathsf{p} \in \Pi$ is a proposition: $\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \mathcal{U} \varphi \mid \bigcirc\varphi$. For convenience, we define the two temporal operators $\Diamond\varphi$ and $\Box\varphi$ as macros $\top\mathcal{U}\varphi$ and $\neg\Diamond\neg\varphi$, respectively, where $\top \equiv \mathsf{p} \vee \neg\mathsf{p}$ denotes universal truth. We will omit the set of propositions "Π" as subscript of $\mathbf{LTL}_\Pi$ if clear from context.

LTL-formulae are interpreted over ω-sequences[4] (infinite words) w over sets of propositions, i.e. $w \in \mathcal{P}(\Pi)^\omega$. We use the same notation for words w as introduced for paths λ, e.g. $w[i]$ and $w[i, \infty]$. We note that a model $\mathfrak{M}$ and a path λ in $\mathfrak{M}$—more precisely the valuation function included in the model—induce such an ω-sequence. However, we give the semantics in more general terms because it shall later prove convenient when relating the setting to mechanism design.

Definition 5 *(Semantics $\models^{\mathbf{LTL}}$).* Let Π be a finite, non-empty set of propositions and $w \in \mathcal{P}(\Pi)^\omega$. The semantics of $\mathbf{LTL}_\Pi$-formulae is given by the satisfaction relation $\models^{\mathbf{LTL}}$ defined by the following cases:

$w \models^{\mathbf{LTL}} \mathsf{p}$ iff $\mathsf{p} \in w[0]$ and $\mathsf{p} \in \Pi$;
$w \models^{\mathbf{LTL}} \neg\varphi$ iff not $w \models^{\mathbf{LTL}} \varphi$ (we write $w \not\models^{\mathbf{LTL}} \varphi$);
$w \models^{\mathbf{LTL}} \varphi \wedge \psi$ iff $w \models^{\mathbf{LTL}} \varphi$ and $w \models^{\mathbf{LTL}} \psi$;
$w \models^{\mathbf{LTL}} \bigcirc\varphi$ iff $w[1, \infty] \models^{\mathbf{LTL}} \varphi$; and
$w \models^{\mathbf{LTL}} \varphi \mathcal{U} \psi$ iff there is an $i \in \mathbb{N}_0$ such that $w[i, \infty] \models \psi$ and $w[j, \infty] \models^{\mathbf{LTL}} \varphi$ for all $0 \leq j < i$.

Given a path λ and a valuation π we also write $\pi(\lambda)$ for $\pi(\lambda[0])\pi(\lambda[1])\ldots$. Moreover, we just write $\lambda \models \varphi$ if the model and its valuation function are clear from context. Similarly, we usually omit **LTL** in $\models^{\mathbf{LTL}}$.

Example 4 *(Preference).* In our scenario, we assume that car 1 has the preference $\mathsf{p}_1^{\mathsf{s}} \wedge \mathsf{p}_2^{\mathsf{s}} \rightarrow \bigcirc\mathsf{p}_1^{\mathsf{e}}$ modelling that it wants the next state to be its ending state (i.e., it prefers to pass through the narrow passage as the first car). The path $q_0 q_2^\omega$ violates the preference, we have: $\pi(q_0 q_2^\omega) \not\models \mathsf{p}_1^{\mathsf{s}} \wedge \mathsf{p}_2^{\mathsf{s}} \rightarrow \bigcirc\mathsf{p}_1^{\mathsf{e}}$ for π being the valuation function of $\mathfrak{M}_1$ from Example 2.

We use preference lists to define preferences of agents [23]. Such a list consists of a sequence of **LTL**-formulae each coupled with a natural number. The formula is a binary classifier of paths in the model—considering the induced ω-words over propositions. The natural number assigns a utility to the paths which satisfy the respective formula. Thus, a preference list assigns a utility value to all paths in a model.

[4] We could also give a semantics over paths in a given model. We use ω-sequences over sets of propositions as it makes the semantics independent of the structure of a specific model which shall prove useful when relating the setting to mechanism design in Section 3.

Definition 6 *(Preference list, preference profile).* A *preference list* over a set of propositions Π of an agent $i \in \mathbb{A}\text{gt} = \{1, \ldots, k\}$ 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 $\varphi_j \in \mathbf{LTL}_\Pi$, each $u_j \in \mathbb{N}$ for $j = 1, \ldots, l-1$, $\varphi_l = \top$, and $u_l = 0$. A *preference profile* over Π and $\mathbb{A}\text{gt}$ is given by $\vec{\gamma} = (\gamma_1, \ldots, \gamma_k)$ containing a preference list over Π for each agent in $\mathbb{A}\text{gt}$. We typically use $\mathcal{P}refs$ to denote a non-empty set of such preference profiles. We omit mentioning the set of propositions Π and the set of agents $\mathbb{A}\text{gt}$, respectively, if clear from context.

A preference list of an agent may be interpreted as the agent's goals, denoting the behaviours that the agent wants to realise. Given an agent $i \in \mathbb{A}\text{gt}$ with preference list $\gamma = ((\varphi_1, u_1), \ldots, (\varphi_l, u_l))$, a word w is assigned utility u_j (to denote the utility of outcome path λ with $\pi(\lambda) = w$ for agent i) if φ_j is the formula with the smallest index j in γ that is satisfied on w, i.e., $w \models \varphi_j$ and for all $l < j$ we have that $w \not\models \varphi_l$. Note that there is always a formula in γ which is satisfied on w since the last formula in a preference list is required to be $\top$.

Example 5 *(Preference profiles).* In our scenario, we consider the following two preference profiles $\vec{\gamma}_1$ and $\vec{\gamma}_2$ of the cars where proposition fine_i should be read as agent i has received a fine.

$$\begin{aligned}\vec{\gamma}_1 = \quad & \{((\bigcirc p_1^e \wedge \Box\neg\mathsf{fine}_1, 3), (\Diamond p_1^e \wedge \Box\neg\mathsf{fine}_1, 2), (\Box\neg\mathsf{fine}_1, 1), (\top, 0)),\\ & ((\bigcirc p_2^e \wedge \Box\neg\mathsf{fine}_2, 3), (\Diamond p_2^e \wedge \Box\neg\mathsf{fine}_2, 2), (\Box\neg\mathsf{fine}_2, 1), (\top, 0))\ \}\end{aligned}$$

$$\begin{aligned}\vec{\gamma}_2 = \quad & \{((\bigcirc(p_1^e \wedge p_2^e) \wedge \Box\neg\mathsf{fine}_1, 3), (\bigcirc p_1^e \wedge \Box\neg\mathsf{fine}_1, 2), (\Box\neg\mathsf{fine}_1, 1), (\top, 0)),\\ & ((\bigcirc(p_1^e \wedge p_2^e) \wedge \Box\neg\mathsf{fine}_2, 3), (\bigcirc p_2^e \wedge \Box\neg\mathsf{fine}_2, 2), (\Box\neg\mathsf{fine}_2, 1), (\top, 0))\ \}\end{aligned}$$

The first preference profile is egoistic in the sense that the cars first prefer to reach their end positions directly without any sanction, then eventually to get to their end positions without any sanction, and finally to get no sanction. The second preference profile is more social because cars now first prefer that they both get to their end positions directly.

We note that technically fine_1 and fine_2 are simply two fresh propositional symbols. As before states in the model can be labelled with them. Thus, it should be intuitive that the preferences can be used to classify paths according to the preferences. Later, in Section 3.1.4 we formally introduce the formal machinery to update a model by such new propositional symbols; in particular, in Example 11 we shall return to a formal treatment in the context of this example.

2.3. From CGS to game theory

In order to analyse the behaviour of agents and the emerging system behaviour, we use the machinery of game theory (cf. [54]). A *strategic game form* is a tuple $\hat{G} = (\mathbb{A}\text{gt}, (Ac_i)_{i\in\mathbb{A}\text{gt}}, O, g)$ where $\mathbb{A}\text{gt}$ is the set of agents, Ac_i is the set of actions available to agent $i \in \mathbb{A}\text{gt}$, O is a set of outcomes, and $g : Ac \to O$ with $Ac = \times_{i\in\mathbb{A}\text{gt}} Ac_i$ is an outcome function that associates an outcome with every action profile. For the sake of readability, we use the same notation for agents in strategic game forms and CGSs. A *strategic game* $G = (\hat{G}, (\succeq_i)_{i\in\mathbb{A}\text{gt}})$ extends a strategic game form with a preference relation $\succeq_i$ on O for each agent $i \in \mathbb{A}\text{gt}$. The preference relation of agent i induces a preference relation on action profiles: $a_1 \succeq_i a_2$ iff $g(a_1) \succeq_i g(a_2)$ for any $a_1, a_2 \in Ac$. We also use payoff functions $\mu_i : O \to \mathbb{R}$ to represent preference relations where higher payoff values correspond to more preferred outcomes.

It is well known how an extensive form game can be transformed to a strategic game without changing specific sets of strategy profiles, e.g. the set of Nash equilibria. The next definition connects CGSs to strategic games following the notation of [23]. The actions of the strategic game correspond to the (memoryless) strategies in the CGS. A payoff function is obtained from preference profiles and the outcome paths resulting from strategy profiles.

Definition 7 *(CGS $\rightsquigarrow$ strategic game form, strategic game).* Let $(\mathfrak{M}, q)$ be a pointed CGS with $\mathbb{A}\text{gt} = \{1, \ldots, k\}$ and $\vec{\gamma} = (\gamma_1, \ldots, \gamma_k) \in \mathcal{P}refs$ be a preference profile. We define $\Gamma(\mathfrak{M}, q)$ as the strategic game form $(\mathbb{A}\text{gt}, (\Sigma_i)_{i\in\mathbb{A}\text{gt}}, \Lambda_\mathfrak{M}(q), g)$ associated with $(\mathfrak{M}, q)$ where $\mathbb{A}\text{gt}$ is the set of agents in $\mathfrak{M}$, Σ_i is the set of strategies of agent i (cf. Definition 2), $\Lambda_\mathfrak{M}(q)$ the set of all paths in $\mathfrak{M}$ starting in state q, and $g(s) = out_\mathfrak{M}(q, s)$ for $s \in \Sigma_{\mathbb{A}\text{gt}}$. Moreover, we define $\Gamma(\mathfrak{M}, q, \vec{\gamma})$ as the strategic game $(\Gamma(\mathfrak{M}, q), (\mu_i)_{i\in\mathbb{A}\text{gt}})$ associated with $(\mathfrak{M}, q)$ and $\vec{\gamma}$, where for all $s \in \Sigma_{\mathbb{A}\text{gt}}$ the payoff function μ_i is defined as follows: $\mu_i(s) = u_j$ where $\gamma_i = ((\varphi_1, u_1), \ldots, (\varphi_{l-1}, u_{l-1}), (\varphi_l, u_l))$ and j is the minimal index such that $\pi(out_\mathfrak{M}(q, s)) \models^{\mathbf{LTL}} \varphi_j$.

In general, not all paths from a CGS can be obtained by memoryless strategies, i.e. $\bigcup_{s\in\Sigma_{\mathbb{A}\text{gt}}} out_\mathfrak{M}(q, s) \neq \Lambda_\mathfrak{M}(q)$. As mentioned before, we use memoryless strategies because of their better computational properties. Note that the obtained strategic game form is always finite as the set of strategies of each agent is finite. Note also that the just defined strategic game is well-defined, especially because for the last entry (φ, u) of each preference list we have that $\varphi = \top$.

Example 6 *(Strategic game form).* For our car scenario, the strategic game form $\Gamma(\mathfrak{M}_1, q_0)$ associated with $(\mathfrak{M}_1, q_0)$ is illustrated in Fig. 3. Strategies are represented according to the conventions of Example 3.

1\2	$M\star_2$	WM	WW
$M\star_1$	$q_0q_3^\omega$	$q_0q_1q_4^\omega$	$q_0q_1^\omega$
WM	$q_0q_2q_4^\omega$	q_0^ω	q_0^ω
WW	$q_0q_2^\omega$	q_0^ω	q_0^ω

Fig. 3. The strategic game form $\Gamma(\mathfrak{M}_1, q_0)$ associated with the pointed CGS $(\mathfrak{M}_1, q_0)$ from Example 2. We have $\star_1, \star_2 \in \{M, W\}$.

$\Gamma(\mathfrak{M}_1, q_0, \vec{\gamma}_1)$

1\2	$M\star_2$	WM	WW
$M\star_1$	1\1	**3\2**	3\1
WM	**2\3**	1\1	1\1
WW	1\3	1\1	1\1

$\Gamma(\mathfrak{M}_1, q_0, \vec{\gamma}_2)$

1\2	$M\star_2$	WM	WW
$M\star_1$	**1\1**	**2\1**	**2\1**
WM	**1\2**	1\1	1\1
WW	**1\2**	1\1	1\1

Fig. 4. The strategic games $\Gamma(\mathfrak{M}_1, q_0, \vec{\gamma}_1)$ and $\Gamma(\mathfrak{M}_1, q_0, \vec{\gamma}_2)$. Again, we have $\star_1, \star_2 \in \{M, W\}$. Payoff profiles given in bold indicate Nash equilibria. We use $x\backslash y$ to denote a payoff of x and y for agent 1 and 2, respectively.

A game theoretic solution concept, e.g. Nash equilibria or dominant strategy equilibrium, can be considered as a function $\mathcal{S}$ the domain of which is the set of strategic games and the image of which is a set of strategy profiles [54]. That is, for each strategic game G with strategy profiles Σ we have that $\mathcal{S}(G) \subseteq \Sigma$. We assume that the reader is familiar with the notion of solution concept and refer to [54] for further details. In the present work we are mainly concerned with the concept of Nash equilibrium and also discuss dominant strategy equilibrium. Therefore, we give the basic definitions. An action profile $(a_1, \ldots, a_k)$, containing an action for each agent, is a *Nash equilibrium* if no agent can unilaterally deviate to get a better payoff; that is, for each agent i it must be the case that for all actions a_i' of that agent it holds that $(a_1, \ldots, a_k) \succeq_i (a_1, \ldots, a_{i-1}, a_i', a_{i+1}, \ldots, a_k)$. We use $\mathcal{NE}(G)$ to refer to the Nash equilibria in a given game G. A dominant strategy equilibrium is a stronger notion. Formally, a profile $(a_1, \ldots, a_k)$ is a *(weakly) dominant strategy equilibrium* if for each agent i, the action a_i is the best action independent of the choices of the opponents. Such an action a_i is called *(weakly) dominant strategy*.[5] We refer to [58] for more details. Furthermore, we use $\mathcal{DOE}(G)$ to refer to the set of dominant strategy equilibria in a given game G. We lift the notion to tuples consisting of a CGS $\mathfrak{M}$ and a preference profile $\vec{\gamma}$ by $\mathcal{S}(\mathfrak{M}, q, \vec{\gamma}) := \mathcal{S}(\Gamma(\mathfrak{M}, q, \vec{\gamma}))$.

Example 7 *(Example 6 contd.: strategic games).* We include the preference profiles $\vec{\gamma}_1$ and $\vec{\gamma}_1$, respectively, from Example 5 in the strategic game form of Example 6. We obtain the strategic games $\Gamma(\mathfrak{M}_1, q_0, \vec{\gamma}_1)$ and $\Gamma(\mathfrak{M}_1, q_0, \vec{\gamma}_2)$ which are shown in Fig. 4. Bold entries are used to identify Nash equilibria. For example, we have that $\mathcal{NE}(\mathfrak{M}_1, q_0, \vec{\gamma}_1) = \{(WM, M\star_2) \mid \star_2 \in \{M, W\}\} \cup \{(M\star_1, WM) \mid \star_1 \in \{M, W\}\}$. The game $\Gamma(\mathfrak{M}_1, q_0, \vec{\gamma}_1)$ has no dominant strategy equilibria whereas $\mathcal{DOE}(\mathfrak{M}_1, q_0, \vec{\gamma}_2) = \{(M\star_1, M\star_2) \mid \star_1, \star_2 \in \{M, W\}\}$.

It is worth to note that pure Nash equilibria as well as dominant strategy equilibria may not exist. The matching pennies game[6] is a classical, strictly-competitive game without Nash equilibria [54].

Concluding remarks

In this section we introduced the formal setting in which we shall study the effects of imposing norms on a multi-agent system. We used **LTL** to define agent's preferences. In combination with concurrent game structures they allow us to relate the multi-agent setting to normal form games, which are a well-studied mathematical model to investigate the outcome of interactions among rational agents. We presented the concept of Nash equilibrium and dominant strategy equilibrium as examples to capture agents' rational behaviour. The general idea of normative mechanism design, introduced in the next section, however, does not depend on a specific solution concept.

3. Norm-based mechanism design

So far we have considered a concurrent game structure as a formal tool to model a multi-agent system. We also assumed that agents have preferences over the system behaviour. As the actual agents' preferences may not be known in advance, we may only assume that a *set* of possible preference profiles over the system behaviour is given—containing those preferences which seem sensible to the system designer. In the previous section we explained how these ingredients set the stage to analyse the "optimal" behaviours of the multi-agent systems by considering the equilibria of the game constructed from the concurrent game structure and the preference profiles. With "optimal" we refer to the system behaviours that correspond

[5] We note that this is a rather weak definition of dominance, other definitions use a strict preference relation $a \succ_i a'$, defined as $a \succeq_i a'$ and not $a' \succeq_i a$.

[6] Each of two agents shows one side of a coin. One agent wins if both coins show the same side. The other wins if this is not the case. The matching pennies game can, e.g., also be found in [54].

Notation	Description	Defined/Used	Page
$\mathbb{A}$gt	set of players	Definition 1	100
Act	set of actions	Definition 1	100
α_i	action of player i, used in the multi-agent setting	Definition 1	100
Q	set of states	Definition 1	100
q	state	Definition 1	100
$\mathfrak{M}$	concurrent game structure	Definition 1	100
Σ	set of (memoryless) strategies	Definition 2	99
Λ	set of paths	Section 2.1	100
out	outcome of strategy	Definition 3	101
s	memoryless strategy	Definitions 2	101
γ_i	preference list of player i	Definition 6	103
$\mathcal{P}refs$	set of preference lists	Definition 6	103
μ_i	utility function of player i	Definition 7	103
O	outcomes	Section 2.3	103
$\succeq_i$	preference relation of player i	Section 2.3	103
a_i	action of player i, used in the game theoretical setting	Section 2.3	103
G, $\hat{G}$	strategic game, strategic game form/mechanism	Section 2.3	103
$\mathcal{G}$	a set of strategic game forms	Section 3.1.1	105
$\mathcal{S}$	a solution concept	Section 3.1.1	105
P	players	Section 3.1.1	105
f	normative behaviour function	Definition 8	106
Π	propositions (hard and soft facts)	Section 3.1.3	106
Π_h	hard facts	Section 3.1.3	106
Π_s	soft facts	Section 3.1.3	106
$\mathcal{A}$	subset of action profiles	Definition 9	107
N	normative system	Definition 9	107
N_ϵ	empty normative system	Definition 9	107
RN	set of regimentation norms	Definition 9	107
SN	set of sanctioning norms	Definition 9	107
$\mathfrak{I}$	implementation setting	Definition 12	112
$\mathcal{N}$	set of normative systems	Definition 12	112
Act_i	set of actions of player i	Definition 15	117

Fig. 5. Overview of commonly used notation.

to the outcome of Nash equilibria. In this section, we further assume that a social choice function is given that models the multi-agent system designer's preferred system behaviour, also called the *social perspective*. A social choice function indicates the system behaviour that is socially preferred (or preferred by the system designer) when the agents act according to a specific preference profile.

The problem that we consider in this section is the *implementation problem* that can be formulated as follows. Suppose that for a given set of agents' preference profiles the socially preferred system behaviour is not aligned with the agents' preferred system behaviour, i.e. for the given agents' preference profiles the optimal system behaviour from the agents' perspective (represented by Nash equilibria) is not aligned/contradicts with the optimal system behaviour from the system designer's perspective (represented by the social choice function). The question is whether the enforcement of some norms (e.g., by means of regimentation and sanctions) on the agents' behaviour can align the preferred system behaviour from both perspectives, i.e., whether the enforcement of some norms on the agents' behaviour can change the agents' behaviour toward the socially preferred system behaviour. A second more elaborate question is about the existence of a set of norms whose enforcement aligns the preferred system behaviour from the agents' as well as system designer's perspectives. We coin the design of a set of norms to change the agents' behaviour, which is inspired by mechanism design, *norm-based mechanism design*.

In order to ease the reading of the rest of this paper, we list some of the concepts used in our formalisation and their notation in Fig. 5.

3.1. Preliminaries

We start with reviewing concepts from classical mechanism design and introducing corresponding concepts for norm-based mechanism design. In particular, we define the concept of normative behaviour function which is the counterpart of the concept of social choice function in classical mechanism design. We then introduce the concept of norms and norm-based multi-agent system, and explain how they can influence the agents' behaviour. We define the notion of norm enforcement formally as an update function that changes the specification of the multi-agent system based on a given set of norms. These concepts set the stage for *norm-based mechanism design*.

3.1.1. Classical mechanism design

In our exposition of classical mechanism design we mostly follow the presentation of [54], in particular we often use their notation which allows us to clearly relate the classical concepts with the corresponding ones of norm-based mechanism design. In social choice theory a *social choice rule* $f : P \to \mathcal{P}(O)$ assigns a subset of outcomes from the set of outcomes

O to a preference profile $\succeq = (\succeq_i)_{i \in \mathbb{A}gt} \in P$, consisting of a preference relation $\succeq_i$ over O for each agent from $\mathbb{A}gt$. Here, we consider P to be a set of such preference profiles. In contrast, a social choice *function* picks exactly one element of the outcome rather than a set of outcomes. For example, a profile $(\succeq_1^{fast}, \succeq_2^{fast})$ could express that both agents have a preference for driving fast. In that case a social choice function can be used to map the preference profile to the outcome o^h representing "build highway", i.e. $f((\succeq_1^{fast}, \succeq_2^{fast})) = o^h$.

In the following discussion we focus on social choice rules. The outcome $f(\succeq)$ is the *social outcome* defined by the social choice rule f. *Mechanism design* is concerned with the problem of constructing a *mechanism*—a strategic game form—which implements a social choice rule assuming a specific rational behaviour of the agents. The mechanism is constructed over a structure fixing the set of agents $\mathbb{A}gt$, the set of possible outcomes O, the set of possible preference profiles P over outcomes O, and a set of strategic game forms $\mathcal{G}$ with outcomes in O. The mechanism is drawn from $\mathcal{G}$. This fixed structure has different names in the literature. Osborne and Rubinstein [54] refer to it as *environment* while Shoham and Leyton-Brown [58] use a related formalisation in terms of Bayesian game settings. We follow [54] but, in order to avoid confusion with our notation, refer to the structure as *implementation setting*. Now, a strategic game form $G \in \mathcal{G}$ together with a preference profile $\succeq \in P$ defines a *strategic game* $(G, \succeq)$. Given a solution concept $\mathcal{S}$ (e.g., Nash equilibrium) for strategic games, i.e. a mapping from $(G, \succeq)$ to a set of action profiles, we can formulate the $\mathcal{S}$-implementation problem over an implementation setting as follows. A game form $G \in \mathcal{G}$ *$\mathcal{S}$-implements the social choice rule f over P* if, and only if, for all preference profiles $\succeq \in P$ and all equilibria $(a_1, \ldots, a_{|\mathbb{A}gt|}) \in \mathcal{S}(G, \succeq)$ we have that the outcome obtained by $(a_1, \ldots, a_{|\mathbb{A}gt|})$ is contained[7] in $f(\succeq)$. We emphasize that $\mathcal{S}(G, \succeq)$ contains the strategy profiles in the game $(G, \succeq)$ that satisfy the solution concept $\mathcal{S}$.

3.1.2. Normative behaviour function

As discussed above, in social choice theory a *social choice rule* assigns outcomes to given profiles of preferences. In our setting, the social choice rule represents the preference of the system designer and is defined as a function[8] that assigns an **LTL**-formula—describing *a set* of paths—to each preference profile (a sequence of sequences of **LTL**-formulae) of the agents. From now on, when considering norm-based mechanism design concepts we use our own notation that has been introduced in previous sections.

Definition 8 *(Normative behaviour function).* Let $\mathcal{P}refs$ be a set of preference profiles over Π. A *normative behaviour function*[8] f is a mapping $f : \mathcal{P}refs \to \mathbf{LTL}_\Pi$.

Similar to classical mechanism design, the preference of the system designer describes the designer's desired outcome if the agents had the given preference profile as their true profile. Thus, representing the preference of the system designer by a (normative behaviour) function allows us to model the system designer's uncertainty about the true preferences of the agents. The following is a simple example where the preference of the system designer is independent of the agents' preferences such that it maps possible preference profiles to an **LTL**-formula. We choose deliberately to keep the example simple, but it should be clear that the preference of the system designer is not always a single formula and often depends on the agents' preferences.

Example 8 *(Normative behaviour function).* We assume that the preference of the system designer is represented by the following normative behaviour function, which indicates that in all cases the first car should reach its end position directly: $f(\vec{\gamma}_i) = \bigcirc p_1^e$.

We refer to the outcome as the *normative outcome wrt. a given preference profile*. In our view, the aim of *norm-based mechanism design* is to come up with a *norm-based 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 possible environment executions stay within the normative outcome. The idea is that norms and sanctions will (de)motivate agents to perform specific actions.

Example 9 *(Non-aligned preferences).* Following Example 7, for $\vec{\gamma}_1$, the path $q_0 q_2 q_4^\omega$ (yielded by strategy profile (WM, MW), which is a Nash equilibrium) in $\mathfrak{M}_1$ does not satisfy the preference $f(\vec{\gamma}_1)$ of the system designer.

3.1.3. Normative system, hard and soft facts

In the remainder of this section, let Π again be a finite and non-empty set of propositions. We assume that Π can be partitioned into two types of propositions: the set Π_h denotes the *hard facts* and Π_s the *soft facts*. Hard facts describe the (physical/brute) properties of the multi-agent environment which in turn are used to define the *action structure* of a CGS.

[7] Sometimes, it is assumed that *all* outcomes specified by $f(\succeq)$ can be obtained by some equilibrium. It is also common to consider a social choice function that maps to single outcomes rather than a social choice rule.

[8] We emphasize its definition as a function. When considering the set of paths satisfying a **LTL**-formula rather than the formula itself, however, it shows the characteristics of a social choice rule.

For example, typical hard facts describe that a car is at a certain position or that the traffic light is red. Soft facts are used to model properties which do not affect the action structure but can affect the agents' preferences. For example, they model that a car has violated a traffic norm, a car received a fine, or a car arrived late at its destination (where we use car as a shorthand for "the driver of a car" etc.). In other words, the modification of soft facts does not change the available actions nor their executability but they can affect the evaluation of the agents' preferences.

The classification into hard and soft facts depends on the specific modelling. In the following we assume that we are given a set of *hard facts* Π_h and a set of *soft facts* Π_s, where sets are finite, non-empty and disjoint. Therefore, we assume that a CGS $\mathfrak{M} = (\mathbb{A}\text{gt}, Q, \Pi, \pi, Act, d, o)$ is given with $\Pi = \Pi_h \cup \Pi_s$. In this paper, we also distinguish between two types of norms. *Sanctioning norms* are enforced through sanctions imposed on the execution of actions from specific states. *Regimenting norms* are enforced by modifying the effect of the execution of actions from a specific state; they make the actions effectless [27]. In the following, we use $\mathbf{PL}_X$ for a set X of propositional atoms to refer to all propositional formulae over X and denote by $\models^{\mathbf{PL}}$ the satisfaction relation of propositional logic.

Definition 9 *(Norms and normative system).* A *norm* (over $\mathfrak{M}$) is an element from $\mathbf{PL}_{\Pi_h} \times \mathcal{P}(Act^k)\backslash\{\emptyset\} \times (\mathcal{P}(\Pi_s)\backslash\{\emptyset\} \cup \{\bot\})$. A norm of type $(\varphi, \mathcal{A}, \bot)$ is a *regimenting norm* and one of type $(\varphi, \mathcal{A}, S)$ with non-empty $S \subseteq \Pi_s$ a *sanctioning norm*. A set of such norms N is called a *normative system* (over $\mathfrak{M}$) with a typical norm denoted by η. $\mathsf{N}_\epsilon = \emptyset$ is the *empty normative system*, and $\mathcal{N}_{\mathsf{rs}} = \{\mathsf{N}_1, \mathsf{N}_2, \ldots\}$ is the set of all normative systems. We define $\mathcal{N}_{\mathsf{s}}$ and $\mathcal{N}_{\mathsf{r}}$ as the set of all normative systems consisting of only sanctioning and regimenting norms, respectively. We often write $(\varphi, \mathcal{A}, \cdot)$ to refer to a sanctioning or regimenting norm.

A norm $(\varphi, \mathcal{A}, \bot)$ should be read as: "it is forbidden to perform action profiles from $\mathcal{A}$ in φ-states (i.e. states in which φ holds) and that the enforcement of this norm prevents action profiles in $\mathcal{A}$ to be performed". A norm $(\varphi, \mathcal{A}, S)$ should be read as: "it is forbidden to perform action profiles from $\mathcal{A}$ in φ-states and that the enforcement of this norm imposes sanctions S on the states resulting by performing the very action profile from $\mathcal{A}$ in a φ-state". The basic idea is that sanctioning norms enforced on a model change the valuation of states by soft facts only; that is, the underlying physical structure represented by hard facts remains intact and thereby the action structure of the model remains intact. As we will see later in this paper, the agents' preferences are also defined on soft facts such that any changes in valuations of states by soft facts may affect agents' rational behaviour. Regimenting norms affect the physical structure as they directly affect the action structure of the model.

We note that a norm $(\varphi, \mathcal{A}, \cdot)$ can be seen as a prohibition ("it is prohibited that any action profile in $\mathcal{A}$ is executed in states satisfying φ") but just as well as a obligation to perform an action profile not in $\mathcal{A}$ in any state satisfying φ. This also corresponds to the duality of obligations and prohibitions: a prohibition of doing an action, is the same as being obliged to not doing the action.

Example 10 *(Norms).* In order to avoid cars to congest in the narrow part of the road we introduce two sanctioning norms. The first norm prohibits the first car to wait in the start position $p_1^s \wedge p_2^s$ when the second car waits as well. The violation of this norm imposes sanction fine_1 (i.e., car 1 is sanctioned). This norm is represented as $(p_1^s \wedge p_2^s, \{(W, W)\}, \{\mathsf{fine}_1\})$. The second norm prohibits the second car to move in the start position. This norm is represented as $(p_1^s \wedge p_2^s, \{(\star, M) \mid \star \in \{M, W\}\}, \{\mathsf{fine}_2\})$. Note that these norms implement a priority for passage in the narrowed road by obliging car 1 to move and car 2 to wait.

It is important to note that sanctioning norms can directly influence the (quantitative) utility that an agent obtains as specific elements of a preference list may no longer be satisfiable. To illustrate this, consider the preference profile $\vec{\gamma}_1$ introduced in Example 5:

$$\vec{\gamma}_1 = \{ \quad ((\bigcirc p_1^e \wedge \Box\neg\mathsf{fine}_1, 3), (\Diamond p_1^e \wedge \Box\neg\mathsf{fine}_1, 2), (\Box\neg\mathsf{fine}_1, 1), (\top, 0))$$
$$((\bigcirc p_2^e \wedge \Box\neg\mathsf{fine}_2, 3), (\Diamond p_2^e \wedge \Box\neg\mathsf{fine}_2, 2), (\Box\neg\mathsf{fine}_2, 1), (\top, 0)) \}$$

In general, there can be an outcome of the system which ensures agent 1 a payoff of 3, i.e. an outcome that satisfies $\bigcirc p_1^e \wedge \Box\neg\mathsf{fine}_1$. The enforcement of a norm by sanctioning norms of type $(\varphi, \mathcal{A}, \{\mathsf{fine}_1\})$ for an appropriate formula φ and set of action profiles $\mathcal{A}$, however, can make it impossible to yield outcomes on which $\neg\Box\mathsf{fine}_1$ holds. In such a case, it would be impossible for the agent to obtain a utility of 3. The technical details of this procedure are introduced in the next section.

3.1.4. Norm-based update

In order to examine the impact of the enforcement of a set of norms on multi-agent systems, we need to determine applicable norms and their sanctions. Therefore, we need to decide which norms are applicable in a concurrent game structure and what are the sanctions that should be imposed on the concurrent game structure. A norm $\eta = (\varphi, \mathcal{A}, \cdot)$ is applicable in a state which satisfies φ and in which an action tuple from $\mathcal{A}$ is being performed.

Definition 10 *(Applicable norms and sanctions).* Let $\mathsf{N} \in \mathcal{N}_{\mathsf{rs}}$ be a normative system, $X \subseteq \Pi$ be a set of facts, and $\vec{\alpha}$ be an action profile. The set of norms from N *applicable* wrt. X and $\vec{\alpha}$, denoted by $\mathsf{App}_{\mathsf{N}}(X, \vec{\alpha})$, is defined as follows:

$$\mathsf{App}_{\mathsf{N}}(X, \vec{\alpha}) = \{\eta \in \mathsf{N} \mid \eta = (\varphi, \mathcal{A}, \cdot) \text{ with } X \models^{\mathbf{PL}} \varphi \text{ and } \vec{\alpha} \in \mathcal{A}\}.$$

The set of sanctions from N that should be imposed based on X and $\vec{\alpha}$, denoted as $\mathsf{San}_{\mathsf{N}}(X, \vec{\alpha})$, is computed as follows:

$$\mathsf{San}_{\mathsf{N}}(X, \vec{\alpha}) = \begin{cases} \bigcup\{S \mid (\varphi, \mathcal{A}, S) \in \mathsf{App}_{\mathsf{N}}(X, \vec{\alpha})\} & \text{if } \mathsf{App}_{\mathsf{N}}(X, \vec{\alpha}) \text{ contains no regimenting norm,} \\ \{\bot\} & \text{otherwise.} \end{cases}$$

We note that the evaluation of $X \models^{\mathbf{PL}} \varphi$ can be done in polynomial time as it corresponds to evaluating a propositional formula with respect to a given truth assignment represented by the set X. Note that we use a set of facts $X \subseteq \Pi$ (rather than a state $q \in Q$) that triggers norms and sanctions. We do this in order to reuse this definition later on in this paper. In the following, we use $\mathsf{San}_{\mathsf{N}}(\pi(q), \vec{\alpha})$ to determine the sanctions wrt. N, q and $\vec{\alpha}$. We emphasize that the computed sanctions in a state q are meant to be imposed on the state which is reached when executing $\vec{\alpha}$, i.e., the computed sanctions are imposed on state $o(q, \vec{\alpha})$.

The enforcement of a set of norms on a multi-agent system can be modelled as updating the concurrent game structure of the multi-agent system by a normative system, i.e. by regimenting or sanctioning behaviour depending on the type of norm.

Definition 11 *(Update by norms).* Let $\mathfrak{M} = (\mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o)$ be a concurrent game structure and $\mathsf{N} \in \mathcal{N}_{\mathsf{rs}}$ be a normative system over $\mathfrak{M}$. The *update of* $\mathfrak{M}$ *with* N, denoted by $\mathfrak{M} \restriction \mathsf{N}$, is the CGS $(\mathbb{A}\mathrm{gt}, Q', \Pi, \pi', Act, d', o')$ where

1. $Q' := Q \cup \{\ (q, S) \mid \exists q' \in Q, \vec{\alpha} \in Act^k : o(q', \vec{\alpha}) = q,\ S = \mathsf{San}_{\mathsf{N}}(\pi(q'), \vec{\alpha}) \text{ and } \{\bot\} \neq S \neq \emptyset\ \}$
2. $\pi'(x) = \begin{cases} \pi(x) & \text{if } x \in Q \\ \pi(q) \cup S & \text{if } x = (q, S) \end{cases}$
3. $d'_i(x) = d_i(q)$ where $x = q$ or $x = (q, S)$, and $i \in \mathbb{A}\mathrm{gt}$
4. $o'(x, \vec{\alpha}) = \begin{cases} (o(q, \vec{\alpha}), S) & \text{if } S = \mathsf{San}_{\mathsf{N}}(\pi(q), \vec{\alpha}) \text{ and } \{\bot\} \neq S \neq \emptyset \\ o(q, \vec{\alpha}) & \text{if } \mathsf{San}_{\mathsf{N}}(\pi(q), \vec{\alpha}) = \emptyset \\ x & \text{otherwise (i.e. } \{\bot\} = \mathsf{San}_{\mathsf{N}}(\pi(q), \vec{\alpha})) \end{cases}$

   for either $x = (q, S')$ with $S' \subseteq \Pi_s$, or $x = q$.

The first item shows how to duplicate states in Q that are sanctioned but not regimented. The second item defines the updated evaluation of (sanctioned) states. The third item ensures that the new (duplicated) states have the same options as their original counterparts. This is due to the fact that sanctions are solely defined on soft propositions not affecting the transition structure. Finally, the fourth item ensures that the outcome of actions from either the states in Q or their (new) duplicates are in accordance with the original model whenever the actions in those states are not regimented; otherwise when regimented, the actions have no effect and loop in the same state. It is important to notice that looping of a regimented action in a state models a situation where the action cannot be performed and should therefore be interpreted as non-performance of the action. In this case the system remains in the same state. The following example illustrates how the effect of actions is determined in an updated model.

Example 11 *(Norm-based update).* The norm-based update of the environment model $\mathfrak{M}_1$, as illustrated in Fig. 2, based on the normative system $\mathsf{N}_1 = \{\ (p_1^s \wedge p_2^s, \{(W, W)\}, \{\mathsf{fine}_1\}),\ (p_1^s \wedge p_2^s, \{(\star, M) \mid \star \in \{M, W\}\}, \{\mathsf{fine}_2\})\ \}$, denoted by $\mathfrak{M}_2 := \mathfrak{M}_1 \restriction \mathsf{N}_1$, results in a new CGS shown in Fig. 6. In particular, the effects of action profiles (M, M), (M, W), and (W, W) in the duplicated state $(q_0, \{\mathsf{fine}_1\})$ from $\mathfrak{M}_1 \restriction \mathsf{N}_1$ are defined as follows:

$$o'((q_0, \{\mathsf{fine}_1\}), (M, M)) = (o(q_0, (M, M)), \{\mathsf{fine}_2\})$$
$$o'((q_0, \{\mathsf{fine}_1\}), (M, W)) = o(q_0, (M, W))$$
$$o'((q_0, \{\mathsf{fine}_1\}), (W, W)) = (q_0, \{\mathsf{fine}_1\})$$

A different norm-based update of the environment model $\mathfrak{M}_1$ by the normative system that consists of only regimenting norms $\mathsf{N}_2 = \{\ (p_1^s \wedge p_2^s, \{(M, M)\}, \bot),\ (p_1^s \wedge p_2^s, \{(W, M)\}, \bot)\ \}$ is illustrated in Fig. 7. It can be observed that the regimentation of these norms does not allow car 2 to move at the starting position. Note that in our system only action profiles can be the subject of norm regimentation. However, this does not mean that individual agents cannot be the subject to norm regimentation. For example, the resulting transition system in Fig. 7 allows car 1 to do a move action in the starting position while it prevents the move action of car 2.

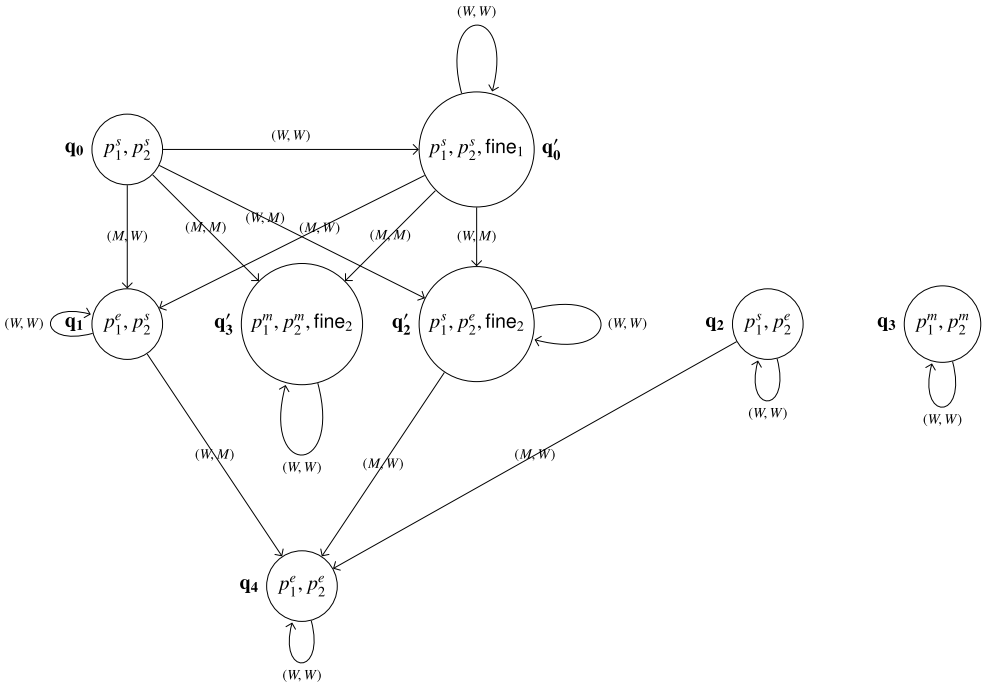

Fig. 6. The CGS $\mathfrak{M}_2$ models $\mathfrak{M}_1 \restriction \mathsf{N}_1$, i.e., the norm-based update of $\mathfrak{M}_1$ based on N_1.

It is important to notice that sanctions do not accumulate through state transitions. This is reflected by the following proposition which expresses that states carry the sanctions caused by the most recent action only, or the most recent action has been regimented.

Proposition 1 *(Non-accumulation of sanctions). Let* $\mathfrak{M} = (\mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o)$ *and* $\mathfrak{M} \restriction \mathsf{N} = (\mathbb{A}\mathrm{gt}, Q', \Pi, \pi', Act, d', o')$, $q \in Q$ *and either* $x = q$ *or* $x = (q, S')$ *with* $S' \subseteq \Pi_s$. *We have*

$$\pi'(o'(x, \vec{\alpha})) \cap \Pi_s = \begin{cases} \mathsf{San}_{\mathsf{N}}(\pi(q), \vec{\alpha}) & \text{if } \mathsf{San}_{\mathsf{N}}(\pi(q), \vec{\alpha}) \neq \{\bot\}, \\ \pi'(x) \cap \Pi_s & \text{otherwise.} \end{cases}$$

Proof. Directly follows from Definition 11 because o' ensures that S' is ignored after $\vec{\alpha}$. □

The non-accumulation of sanctions holds for state transitions that take place based on non-regimented actions. A regimented action that loops in a state with sanctions causes state transitions through which sanctions are not removed (consecutive states through regimentation actions are identical). Although this may suggest that sanctions are accumulated through consecutive states, the persistence of the sanctions in state transitions caused by regimented actions should not be interpreted as receiving sanctions repeatedly. Note that this is consistent with our interpretation that regimented actions cannot be performed, i.e., consecutive states through regimented actions should be considered as remaining in one and the same state. However, we also agree that an alternative solution could be the one where the regimented norm causes a state transition to a fresh copy of the current state with all sanctions removed. This is a design choice. We also note that in a previous version of this work [19] we completely removed specific transitions. As a result the selection of individual actions by agents could yield invalid action profiles. The selection of such invalid action profiles was then essentially avoided by assigning to them a negative utility.

We also observe that updating a CGS with only regimenting norms does not duplicate any state and may only modify the accessibility relation between the state. However, updating a CGS with sanctioning norms can duplicate and introduce new states. This update is essentially different from norm update as introduced by [1]. In case of regimentation we remove accessibility relation between states (by enforcing a looping transition) and in case of sanctioning we add new copies of states and extend the accessibility relation. In [1] an update results in restricting the accessibility relation by removing transitions from the model. Another characteristic of our model update is that different orders of updates with regimenting norms result in the same outcome, which in turn is the same as the single update with the union of regimenting norms. That is, we have that:

$$(\mathfrak{M} \restriction \mathsf{RN}) \restriction \mathsf{RN}' = (\mathfrak{M} \restriction \mathsf{RN}') \restriction \mathsf{RN} = \mathfrak{M} \restriction (\mathsf{RN} \cup \mathsf{RN}')$$

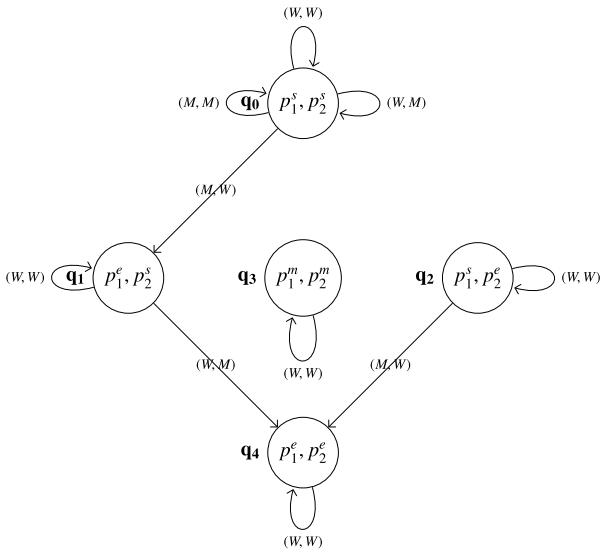

Fig. 7. Norm-based update of $\mathfrak{M}_1$ based on N_2.

Observe that this is not true for sanctioning norms, i.e., different orders of updates with sanctioning norms do not necessarily result in the same CGS. This means that for some CGS $\mathfrak{M}$ and non-empty sets SN_1 and SN_2 we may have

$$(\mathfrak{M} \restriction \mathsf{SN}_1) \restriction \mathsf{SN}_2 \neq (\mathfrak{M} \restriction \mathsf{SN}_2) \restriction \mathsf{SN}_1 \neq \mathfrak{M} \restriction (\mathsf{SN}_1 \cup \mathsf{SN}_2).$$

This is due to the fact that updates of sanctioning norms may duplicate states such that subsequent updates generate different states. One obvious reason different orders of updates by sanctioning norms result in different models is of a syntactic nature, interpreting $=$ as strict/syntactic equality. We illustrate this with a small example. Suppose that a new state (q, S_1) is the result of a norm update. Now, if the model is updated by another normative system, this may yield states of type $((q, S_1), S_2)$. Clearly, reversing the order in which the update is performed yields states of type $((q, S_2), S_1)$ which are different from the previous types of states whenever $S_1 \neq S_2$. Next we consider an example which shows that subsequent updates with sanctioning norms may generate more states than updating with the union of the sanctioning norms at once as illustrated in the example below. Conceptually, all norm violations occurring at the same time should also be sanctioned immediately. This is reflected in our definition of norm update and illustrated in Example 12 and Fig. 8.

Example 12 *(Update order with sanctioning norms).* We consider the CSG $\mathfrak{M}$ shown in Fig. 8 and update it with $\mathsf{SN}_1 = \{(\hat{q}, \{a\}, \{s_1\})\}$ and $\mathsf{SN}_2 = \{(\hat{q}, \{a\}, \{s_2\})\}$ in different orders, and at once ($\hat{q}$ is a propositional formula that is true only in state $\mathbf{q}$), respectively.

The example showed that the models are different. Interestingly, however, in Example 12 if the models $(\mathfrak{M} \restriction \mathsf{SN}_1) \restriction \mathsf{SN}_2$, $(\mathfrak{M} \restriction \mathsf{SN}_2) \restriction \mathsf{SN}_1$ and $\mathfrak{M} \restriction (\mathsf{SN}_1 \cup \mathsf{SN}_2)$ are restricted to states reachable from q, then these models are identical apart from the names of their states. Essentially, given a pointed CGS, different orders of updates with sanctioning norms result in similar CGSs if we consider the parts of the updated CGSs that are reachable from the initial state only. Indeed, it can be shown that the models $(\mathfrak{M} \restriction \mathsf{SN}_1) \restriction \mathsf{SN}_2$ and $(\mathfrak{M} \restriction \mathsf{SN}_2) \restriction \mathsf{SN}_1$ are identical apart from the names of states[9] if restricted to the part reachable from a given state. One may be tempted to conclude that in the same sense these models are similar to $\mathfrak{M} \restriction (\mathsf{SN}_1 \cup \mathsf{SN}_2)$; however, this is not always the case as the reachable part of the latter model may have strictly less states than the previous models. It is our conjecture, however, that the reachable part of these models are bisimilar to each other. As these results are not directly relevant for the exposition of this paper, we omit a formal treatment of this matter.

The order of updates for some non-empty sets of sanctioning and regimenting norms is important as well, i.e., updating first with sanctioning norms and then with regimenting norms may result in different outcomes than updating first with regimenting norms and then with sanctioning norms. Moreover, subsequent updates with regimenting and sanctioning norms do not necessarily result in the same outcome as the single update with the union of regimentation and sanctioning norms. The reason for this is that two consecutive updates by RN and SN result in different outcomes as one single update by $\mathsf{RN} \cup \mathsf{SN}$. The basic intuition is that regimenting norms have priority and if an action has been regimented it does not make sense to sanction the action as it can by the nature of regimentation no longer be executed. This means that for some CGS $\mathfrak{M}$ and non-empty sets SN and RN, we have

[9] More formally, they are isomorphic in the sense of Definition 24. Reachability is formally introduced in Definition 23.

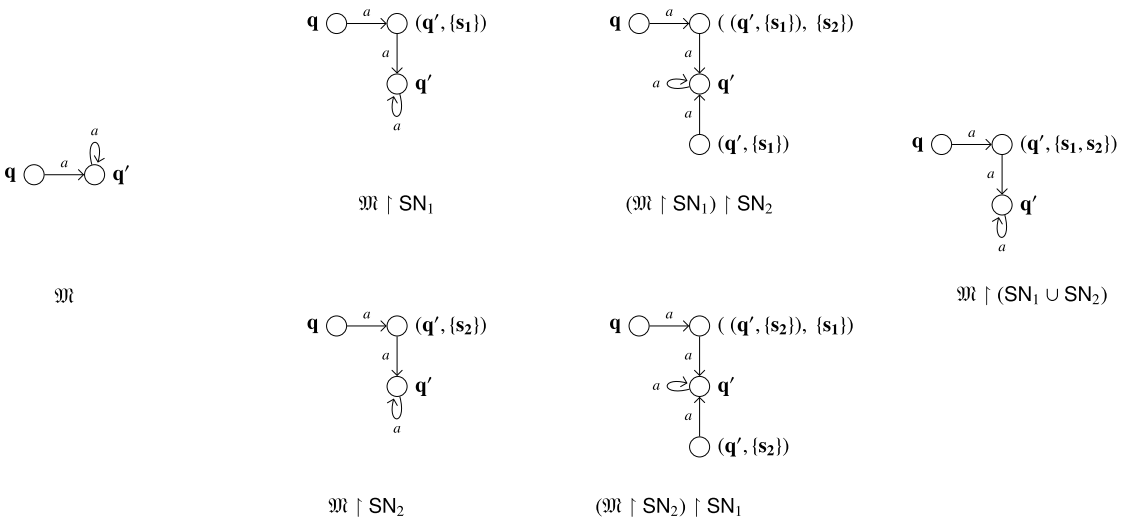

Fig. 8. Updating CGS $\mathfrak{M}$ with sanctioning norms in different orders results in different models.

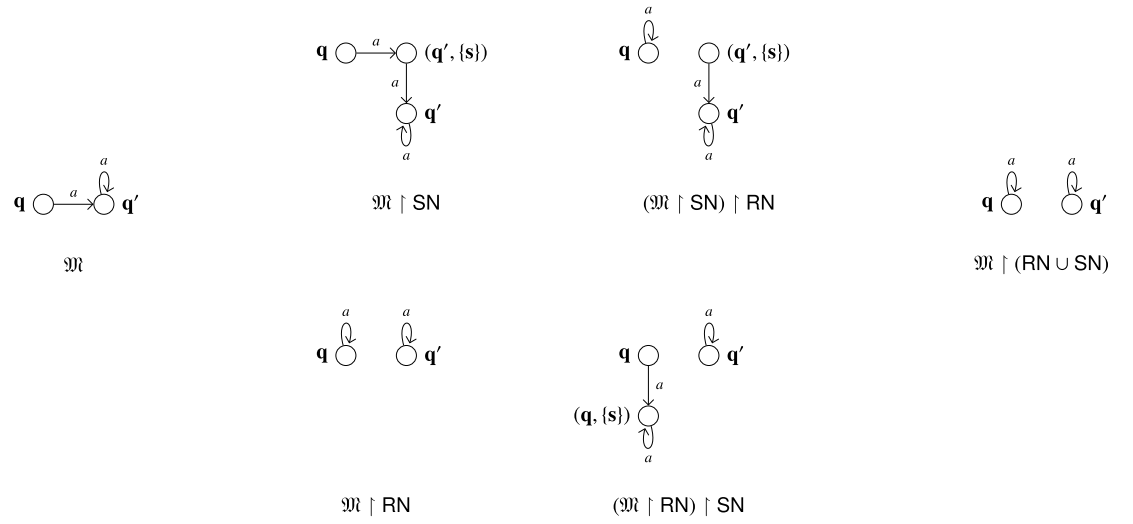

Fig. 9. Updating CGS $\mathfrak{M}$ with sanctioning and regimenting norms in different orders results in different models.

$$(\mathfrak{M} \restriction \mathsf{RN}) \restriction \mathsf{SN} \;\neq\; (\mathfrak{M} \restriction \mathsf{SN}) \restriction \mathsf{RN} \neq \mathfrak{M} \restriction (\mathsf{RN} \cup \mathsf{SN})$$

These observations are illustrated in Example 13 and Fig. 9, where $(\mathfrak{M} \restriction \mathsf{SN}) \restriction \mathsf{RN}$ is different from $(\mathfrak{M} \restriction \mathsf{RN}) \restriction \mathsf{SN}$, which is again different from $\mathfrak{M} \restriction \mathsf{RN}$, which is in this particular case the same as $\mathfrak{M} \restriction (\mathsf{RN} \cup \mathsf{SN})$.

Example 13 *(Update order with sanctioning and regimenting norms).* We consider the CGS $\mathfrak{M}$ shown in Fig. 9 and update it with $\mathsf{SN} = \{(\hat{q}, \{a\}, \{s\})\}$ and $\mathsf{RN} = \{(\hat{q}, \{a\}, \bot)\}$ in different orders, and at once ($\hat{q}$ is a propositional formula that is true only in state **q**), respectively.

We summarize the results in the following proposition.

Proposition 2 *(Properties of norm update). Let* $\mathfrak{M} = (\mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o)$ *be a CGS,* $\mathsf{RN}, \mathsf{RN}' \in \mathcal{N}_\mathsf{r}$ *be sets of regimenting norms, and* $\mathsf{SN} \in \mathcal{N}_\mathsf{s}$ *be a set of sanctioning norms over* $\mathfrak{M}$*. Also let* $\mathsf{N} = \mathsf{RN} \cup \mathsf{SN}$*. Then, we have*

1. $\mathfrak{M} \restriction \mathsf{N}_\epsilon = \mathfrak{M}$
2. $(\mathfrak{M} \restriction \mathsf{RN}) \restriction \mathsf{RN}' \;=\; (\mathfrak{M} \restriction \mathsf{RN}') \restriction \mathsf{RN} \;=\; \mathfrak{M} \restriction (\mathsf{RN} \cup \mathsf{RN}')$
3. *In general, we have:* $(\mathfrak{M} \restriction \mathsf{SN}) \restriction \mathsf{RN} \neq \mathfrak{M} \restriction \mathsf{N} = \mathfrak{M} \restriction (\mathsf{RN} \cup \mathsf{SN}) \neq (\mathfrak{M} \restriction \mathsf{RN}) \restriction \mathsf{SN}$
4. *If forall* $q \in Q$ *and for all* $\vec{\alpha} \in d_{\mathbb{A}\mathrm{gt}}(q)$ *we have that* $\mathsf{App}_{\mathsf{RN}}(\pi(q), \vec{\alpha}) = \emptyset$ *or* $\mathsf{App}_{\mathsf{SN}}(\pi(q), \vec{\alpha}) = \emptyset$ *then* $\mathfrak{M} \restriction \mathsf{N} = (\mathfrak{M} \restriction \mathsf{RN}) \restriction \mathsf{SN}$.

classical mechanism design setting	*norm-based mechanism design setting*
agents $\mathbb{A}$gt	agents $\mathbb{A}$gt
outcomes O	ω-words $\mathcal{P}(\Pi_s \cup \Pi_h)^\omega$
preference relation $\succeq_i$	preference list γ_i over **LTL**-formulae
set of preference profiles P	set of preference profile lists $\mathcal{P}refs$
mechanism $G \in \mathcal{G}$	normative mechanism $(\mathfrak{M}, \mathsf{N})$
set of mechanisms/strategic game forms $\mathcal{G}$	set of norm-based mechanisms $\mathcal{N} = \{\mathsf{N}_1, \mathsf{N}_2, \ldots\}$ wrt. $\mathfrak{M}$
implementation setting $(\mathbb{A}\text{gt}, O, P, \mathcal{G})$	norm-based implementation setting $(\mathfrak{M}, \mathcal{P}refs, \mathcal{N})$
social choice rule $f : P \to \mathcal{P}(O)$	normative behaviour function $f : \mathcal{P}refs \to$ **LTL**

Fig. 10. Correspondence between mechanism design and norm-based mechanism design.

Proof. (1) Obvious. (2) Regimenting norms may introduce loops. Applying a regimenting norm on an already regimented action has no effect. (3) Follows from Example 13. (4) If in no state a regimenting norm and a sanctioning norm are applicable, then the updates cannot interfere. The result follows trivially. □

The properties above concern the iterated applications of the update operation. In the rest of this paper, we focus on single updates.

3.2. Implementation setting

In this section we introduce *norm-based mechanism design* by considering a mechanism using norms to influence the agents' behaviour. We use a normative behaviour function to assign a set of "desired" environment executions (desired from the system designer's perspective), represented by an **LTL**-formula, to each preference profile of the agents. Thus, based on the preferences of the agents the system designer prefers specific outcomes. The aim of *norm-based mechanism design* is to yield a *normative system* (i.e., a set of regimenting and sanctioning norms) for a given environment such that the enforcement of this normative system in the environment motivates the agents to behave in such a way that the outcomes preferred by the system designer are realised. The idea is that the enforcement of norms (de)motivates or prevents agents to perform specific actions. In order to predict how agents act we assume that they act rationally. What playing rationally means shall be specified by game theoretic solution concepts.

Below we introduce the norm-based implementation setting, where mechanism design is interpreted in terms of norms. The basic structure consists of a CGS, a set of preference profiles, and a set of normative systems which if applied on the CGS change the underlying CGS. A tuple consisting of the CGS and such a normative system is called norm-based mechanism. In this sense a norm-based mechanism relates to game forms which are mechanisms in classical mechanism design. Outcomes are defined as paths and preferences over those paths are specified by preference profiles based on **LTL**-formulae. In Fig. 10 we summarize the correspondence between (classical) mechanism design and norm-based mechanism design.

Definition 12 *(Norm-based implementation setting).* A *(norm-based) implementation setting* (NIS) is given by a tuple $\mathfrak{I} = (\mathfrak{M}, \mathcal{P}refs, \mathcal{N})$ where

- $\mathfrak{M} = (\mathbb{A}\text{gt}, Q, \Pi, \pi, Act, d, o)$ is a CGS with $\Pi = \Pi_h \cup \Pi_s$ partitioned into hard and soft facts.
- $\mathcal{P}refs$ is a set of preference profiles over $\mathbb{A}$gt and Π.
- $\mathcal{N} = \{\mathsf{N}_1, \mathsf{N}_2, \ldots\}$ is a set of normative systems over Π with $\mathsf{N}_i = \mathsf{SN}_i \cup \mathsf{RN}_i$, where $\mathsf{SN}_i \in \mathcal{N}_\mathsf{s}$ and $\mathsf{RN}_i \in \mathcal{N}_\mathsf{r}$.

The tuple $(\mathfrak{I}, q)$ where q is a state of $\mathfrak{M}$ is called *pointed (norm-based) implementation setting.*

For the remainder of this section we assume that $\mathfrak{I} = (\mathfrak{M}, \mathcal{P}refs, \mathcal{N})$ is a fixed implementation setting where $\mathfrak{M} = (\mathbb{A}\text{gt}, Q, \Pi, \pi, Act, d, o)$ and q is a state in $\mathfrak{M}$.

Definition 13 *(Norm-based mechanism).* Let $\mathsf{N} \in \mathcal{N}_\mathsf{rs}$ be a *normative system* (over $\mathfrak{M}$). The tuple $(\mathfrak{M}, \mathsf{N})$ is called *norm-based mechanism.*

A norm-based mechanism $(\mathfrak{M}, \mathsf{N})$ gives rise to a CGS by updating $\mathfrak{M}$ with N as introduced in Definition 11.

Example 14 *(Norm-based mechanism).* The environment model $\mathfrak{M}_1$, as illustrated in Fig. 2, together with the normative system N_1, as defined in Example 11, constitute a norm-based mechanism $(\mathfrak{M}_1, \mathsf{N}_1)$, which gives rise to CGS $\mathfrak{M}_2$ from Example 11 and Fig. 6. The CGS $\mathfrak{M}_2$ yields in turn the strategic game form $\Gamma(\mathfrak{M}_1 \upharpoonright \mathsf{N}_1, q_0)$ as illustrated in Fig. 11. For simplicity reasons, we encode strategies as triples (x, y, z). For agent 1 (resp. agent 2) the triple encodes that the agent executes action x at q_0, y at q_0', and z at q_2' (resp. x at q_0, y at q_0', and z at q_1).

A norm-based mechanism defines regimenting and sanctioning norms that influence agents' behaviour in certain directions. Agents are assumed to act rationally, in particular in accordance to their preferences. We use the concept of Nash

$1\backslash 2$	$M \star_2 \star_2'$	WWW	WWM	WMW	WMM
$M \star_1 \star_1'$	$q_0 q_3'^{\omega}$	$q_0 q_1^{\omega}$	$q_0 q_1 q_4^{\omega}$	$q_0 q_1^{\omega}$	$q_0 q_1 q_4^{\omega}$
WWW	$q_0 q_2'^{\omega}$	$q_0 q_0'^{\omega}$	$q_0 q_0'^{\omega}$	$q_0 q_0' q_2'^{\omega}$	$q_0 q_0' q_2'^{\omega}$
WWM	$q_0 q_2' q_4^{\omega}$	$q_0 q_0'^{\omega}$	$q_0 q_0'^{\omega}$	$q_0 q_0' q_2' q_4^{\omega}$	$q_0 q_0' q_2' q_4^{\omega}$
WMW	$q_0 q_2'^{\omega}$	$q_0 q_0' q_1^{\omega}$	$q_0 q_0' q_1 q_4^{\omega}$	$q_0 q_0' q_3'^{\omega}$	$q_0 q_0' q_3'^{\omega}$
WMM	$q_0 q_2' q_4^{\omega}$	$q_0 q_0' q_1^{\omega}$	$q_0 q_0' q_1 q_4^{\omega}$	$q_0 q_0' q_3'^{\omega}$	$q_0 q_0' q_3'^{\omega}$

Fig. 11. The game form associated to $\mathfrak{M}_1 \restriction \mathsf{N}_1$ where $\star_1, \star_1', \star_2, \star_2' \in \{M, W\}$.

$\Gamma(\mathfrak{M}_1 \restriction \mathsf{N}_1, q_0, \vec{\gamma}_1)$

$1\backslash 2$	$M \star_2 \star_2'$	WWW	WWM	WMW	WMM
$M \star_1 \star_1'$	$1\backslash 0$	$3\backslash 1$	$\mathbf{3\backslash 2}$	$3\backslash 1$	$\mathbf{3\backslash 2}$
WWW	$1\backslash 0$	$0\backslash 1$	$0\backslash 1$	$0\backslash 1$	$0\backslash 0$
WWM	$2\backslash 0$	$0\backslash 1$	$0\backslash 1$	$0\backslash 0$	$0\backslash 0$
WMW	$1\backslash 0$	$0\backslash 1$	$0\backslash 2$	$0\backslash 0$	$0\backslash 0$
WMM	$2\backslash 0$	$0\backslash 1$	$0\backslash 2$	$0\backslash 0$	$0\backslash 0$

$\Gamma(\mathfrak{M}_1 \restriction \mathsf{N}_1, q_0, \vec{\gamma}_2)$

$1\backslash 2$	$M \star_2 \star_2'$	WWW	WWM	WMW	WMM
$M \star_1 \star_1'$	$1\backslash 0$	$2\backslash 1$	$\mathbf{2\backslash 2}$	$2\backslash 1$	$\mathbf{2\backslash 2}$
WWW	$1\backslash 0$	$0\backslash 1$	$0\backslash 1$	$0\backslash 1$	$0\backslash 0$
WWM	$2\backslash 0$	$0\backslash 1$	$0\backslash 1$	$0\backslash 0$	$0\backslash 0$
WMW	$1\backslash 0$	$0\backslash 1$	$0\backslash 2$	$0\backslash 0$	$0\backslash 0$
WMM	$2\backslash 0$	$0\backslash 1$	$0\backslash 2$	$0\backslash 0$	$0\backslash 0$

Fig. 12. Strategic games $\Gamma(\mathfrak{M}_1 \restriction \mathsf{N}_1, q_0, \vec{\gamma}_1)$ and $\Gamma(\mathfrak{M}_1 \restriction \mathsf{N}_1, q_0, \vec{\gamma}_2)$ where bold entries mark the Nash equilibria, where $\star_1, \star_1', \star_2, \star_2' \in \{M, W\}$.

equilibrium as rationality criterion. As the fulfilment of agents' preferences depends on the valuation of states on a path, agents' behaviour may be affected if imposing norms and sanctions would cause a modification of states' valuations. Thus, norms can provide an incentive (e.g. in the case of sanctioning norms) to agents to change their behaviour. Therefore, a key question is how to engineer good incentives. In [19], we required that for a successful implementation of a normative behaviour function, all resulting Nash equilibria have to agree with the behaviour specified by the normative behaviour function, which is essentially the case in the classical game theoretic implementation setting given by [54]. Following [69] and [30] we also introduce a weaker notion of implementation where the existence of *some* good behaviour is sufficient. We use the notation of [69] and [30] and introduce weak and strong implementation.

Definition 14 *($\mathcal{S}$-implementation).* Let $\mathfrak{I} = (\mathfrak{M}, \mathit{Prefs}, \mathcal{N})$ and $\mathsf{N} \in \mathcal{N}$. We say that a norm-based mechanism $(\mathfrak{M}, \mathsf{N})$ *(strongly) $\mathcal{S}$-implements the normative behaviour function f over $(\mathfrak{I}, q)$* iff for all $\vec{\gamma} \in \mathit{Prefs}$, $\mathcal{S}(\Gamma(\mathfrak{M} \restriction \mathsf{N}, q, \vec{\gamma})) \neq \emptyset$ and for all $s \in \mathcal{S}(\Gamma(\mathfrak{M} \restriction \mathsf{N}, q, \vec{\gamma})) : out_{\mathfrak{M}\restriction\mathsf{N}}(q, s) \models^{\mathsf{LTL}} f(\vec{\gamma})$. We say that $(\mathfrak{M}, \mathsf{N})$ weakly $\mathcal{S}$-implements normative behaviour function f over $(\mathfrak{I}, q)$ iff for all $\vec{\gamma} \in \mathit{Prefs}$ there is an $s \in \mathcal{S}(\Gamma(\mathfrak{M} \restriction \mathsf{N}, q, \vec{\gamma}))$ such that $out_{\mathfrak{M}\restriction\mathsf{N}}(q, s) \models^{\mathsf{LTL}} f(\vec{\gamma})$. We define the following corresponding sets:

$$\mathsf{SI}_{\mathcal{S}}(\mathfrak{I}, q, f) = \{\mathsf{N} \in \mathcal{N} \mid (\mathfrak{M}, \mathsf{N}) \text{ strongly } \mathcal{S}\text{-implements } f \text{ over } (\mathfrak{I}, q)\}.$$

$$\mathsf{WI}_{\mathcal{S}}(\mathfrak{I}, q, f) = \{\mathsf{N} \in \mathcal{N} \mid (\mathfrak{M}, \mathsf{N}) \text{ weakly } \mathcal{S}\text{-implements } f \text{ over } (\mathfrak{I}, q)\}.$$

If $\mathsf{SI}_{\mathcal{S}}(\mathfrak{I}, q) \neq \emptyset$ (resp. $\mathsf{WI}_{\mathcal{S}}(\mathfrak{I}, q) \neq \emptyset$), we say that f is strongly (resp. weakly) $\mathcal{S}$-implementable over $(\mathfrak{I}, q)$.

The next example gives a norm-based mechanism which strongly $\mathcal{NE}$-implements a normative behaviour function.

Example 15 *(Norm implementation).* Adding the preference profiles $\vec{\gamma}_1$ and $\vec{\gamma}_2$ to the game form $\Gamma(\mathfrak{M}_1 \restriction \mathsf{N}_1, q_0)$ from of Example 14 results in the strategic games shown in Fig. 12. Bold payoffs indicate the Nash equilibria. The equilibria paths do now satisfy the system designer's preference.

Concluding remarks

This concludes our formal model of norm-based mechanism design. The complexity results in the following section are based on Nash equilibrium implementability. In general, other solution concepts, like dominant strategy equilibria, may be considered, depending on the problem at stake. A natural question is why we should be interested in strong or weak implementability. The short answer is: to obtain a stable and desired system behaviour. For illustration let us consider

smart energy grids involving actors such as producers, consumers and regulation authorities. For the sake of the argument we simply assume producers and consumers to act rationally according to the concept of Nash equilibrium. Clearly, this is an idealised and abstracted view,[10] but we believe it delivers the main ideas behind our framework. The regulation authority may have information about the producers' and consumers' preferences. The information might be obtained based on common knowledge (power should always be available), previous behaviour of the producers and consumers, or market research techniques. The regulation authority has some objectives such as flattening peak energy demand by consumers or efficient energy production by producers that it wants to have met. As all actors are self interested, the system (1) may not show stable behaviour in the sense that there is no optimal (social) solution, or (2) may not satisfy the objectives of the regulation authority. In order to overcome these problems, the authority can introduce and enforce norms in order to restrict or incentivize consumers and producers to change their behaviour. For example, additional fees[11] may be charged on consumers if energy is used at specific hours, or fines are imposed on producers for overproduction. In this context, strong implementability means that all stable behaviours satisfy the authority's objectives, i.e., if all actors act rationally and play equilibrium strategies the objectives of the regulation authority are guaranteed. On the other hand, weak implementability indicates that there are some stable behaviour that satisfy the objectives of the regulation authority, but additional means are needed to coordinate on such a behaviour. Weak implementability can therefore be seen as a first step. Obviously, there are many applications for which our formal methodology can be used to analyse and improve the overall system behaviour, e.g., transportation systems, traffic, financial markets and business processes. In general, our formal methodology can be applied to applications where the behaviour of involved actors/processes need to be monitored and controlled. In the next paragraph we shall investigate the complexity of weak and strong implementation.

4. Verification and complexity issues

In this section we consider the complexity of the problem whether some normative system implements a normative behaviour function and whether such a normative system exists at all. We focus on implementation in Nash equilibria. These results are important in order to check whether a norm-governed environment ensures that the agents' rational behaviour satisfies the system specification. We present the proofs of the main results and moved technical details to the appendix.

For the complexity results we first need to be clear about how the size of the input is measured. The size of a CGS is defined as the product of the number of states and the number of transitions, which is denoted by $|\mathfrak{M}|$. The size of a finite normative system N is given by the number of norms it is comprised of times the maximal length propositional formula contained in a norm of N (i.e. a norm is only measured in terms of the size of its precondition, as it can be assumed that the other elements are bound by the size of the model and the agents' preferences). The size of a set of normative systems is measured by $|\mathcal{N}| \cdot |\mathsf{N}_{max}|$ where $\mathsf{N}_{max} \in \mathcal{N}$ is a normative system of maximal size contained in $\mathcal{N}$. Moreover, we assume that any considered normative behaviour function f is polynomial-time computable. The first result investigates the complexity of performing a norm-based update.

Proposition 3. *Let $\mathfrak{I} = (\mathfrak{M}, \mathcal{P}refs, \mathcal{N})$ be given. For any $\mathsf{N} \in \mathcal{N}$, $\mathfrak{M} \restriction \mathsf{N}$ can be computed in polynomial time wrt. the size of $\mathfrak{M}$ and N. The size of $\mathfrak{M} \restriction \mathsf{N}$ is bounded by $O(|\mathfrak{M}|^4)$.*

Proof. Let $\mathfrak{M}' := \mathfrak{M} \restriction \mathsf{N}$, and n and t denote the number of states and transitions in $\mathfrak{M}$, respectively. The set $\mathsf{App}_{\mathsf{N}}(\pi(q), \vec{\alpha})$ can be computed in polynomial time in the size of N: for each $\eta = (\varphi, \mathcal{A}, \cdot) \in \mathsf{N}$ it has to be checked whether $\pi(q) \models^{\mathbf{PL}} \varphi$. To determine the size of $\mathfrak{M}'$ we inspect Definition 11. The number of states of $\mathfrak{M}'$ is bounded by $n + n \cdot n \cdot t = O(n^2 t)$ as the number of tuples $(\pi(q'), \vec{\alpha})$ is bounded by $n \cdot t$ and by Proposition 1 sanctions are not accumulated, i.e. there are at most $n^2 \cdot t$ many new states of the form (q, S). In each new state (q, S) the choices are the same as in q, thus also the number of outgoing transitions. The number of transitions in $\mathfrak{M}'$ is bounded by $O(n^2 t) \cdot t = O(n^2 t^2)$. Thus, the size of $\mathfrak{M}'$ is bounded by $O(n^4 \cdot t^3) \leq O(|\mathfrak{M}|^4)$. □

The next result is concerned with the special case of the weak implementation problem. We investigate the complexity of verifying whether a given normative system weakly implements a given normative behaviour function. Hardness is shown by a reduction of **QSAT**$_2$ which refers to the satisfiability problem of Quantified Boolean Formulae (cf. Appendix A). In the appendix we show how to construct a model $\mathfrak{M}^{\phi}$ and a preference profile from a **QSAT**$_2$-formula ϕ such that there is a Nash equilibrium iff ϕ is satisfiable. The construction of $\mathfrak{M}^{\phi}$ is based on a construction proposed by [22].

[10] In particular, we assume that the amount of energy demanded and supplied can be measured qualitatively expressing, e.g., 'low', 'medium', and 'high' demand.

[11] We note that fines and incentives can be measured quantitatively, assuming that the set of possible numerical values is finite and fixed in advance. We stress that the purpose of this example is to illustrate the conceptual idea of this approach rather than giving a full-fledged real-world modelling of smart grids.

Theorem 4 *(Weak implementation: verification). Let $(\mathfrak{I}, q)$ be a pointed NIS, where $\mathfrak{I} = (\mathfrak{M}, \mathcal{P}refs, \mathcal{N})$ and $\mathsf{N} \in \mathcal{N}$ be a normative system. Let also f be a normative behaviour function. The problem whether $\mathsf{N} \in \mathsf{WI}_{\mathcal{NE}}(\mathfrak{I}, q, f)$ is $\mathbf{\Sigma_2^P}$-complete in the size of $\mathfrak{M}$, N and $\mathcal{P}refs$.*

Proof. *Membership:* We construct a non-deterministic polynomial time oracle Turing machine M which can make calls to a non-deterministic polynomial time Turing machine M'. We first describe the machine M'. It takes as input a strategy profile s and returns true if the profile is not a Nash equilibrium. Therefore, for each agent i, the oracle machine M' guesses an individual strategy s_i' for i and checks whether (s_{-i}, s_i') yields a better payoff than s. If the profile yields a better payoff the machine returns true. The machine M works as follows. Firstly, M computes $\mathfrak{M} \restriction \mathsf{N}$. Then, for each profile $\vec{\gamma} \in \mathcal{P}refs$ the machine guesses a strategy profile s and checks whether $\pi(out(q,s)) \models^{\mathbf{LTL}} f(\vec{\gamma})$. The latter check can be done in polynomial deterministic time as s determines a cyclic path in $\mathfrak{M}$ on which the truth of a linear-time temporal formula can easily be determined. If successful the machine checks whether s is a Nash equilibrium in $\Gamma(\mathfrak{M}, q, \vec{\gamma})$ by calling M' and reverting the answer. This shows that the problem whether $\mathsf{N} \in \mathsf{WI}_{\mathcal{NE}}(\mathfrak{I}, q, f)$ is in $\mathbf{\Sigma_2^P} = \mathbf{NP^{NP}}$.

Sketch of hardness: The hardness result is proven in Theorem 18 in the appendix. Here, we only give the high level idea of the reduction of $\mathbf{QSAT}_2$ to the weak implementability problem. Given a $\mathbf{QSAT}_2$ formula ϕ we construct a two player CGS $\mathfrak{M}^{\phi}$ consisting of the *refuter player* $\mathbf{r}$ and the *verifier player* $\mathbf{v}$. Essentially, the players decide on the value of universally and existentially controlled variables, respectively. Once they decide on a truth assignment, they play a game to evaluate the Boolean formula contained in ϕ following the game theoretical semantics of propositional logic. That is, the *refuter player* $\mathbf{r}$ and the *verifier player* $\mathbf{v}$ control conjunctions and disjunctions, respectively. Using this result, we construct a preference profile $(\gamma_{\mathbf{v}}, \gamma_{\mathbf{r}})$ such that ϕ is satisfiable iff $\exists s \in \mathcal{NE}(\mathfrak{M}^{\phi}, q_0, (\gamma_{\mathbf{v}}, \gamma_{\mathbf{r}}))$ such that $out(q_0, s)$ satisfies a constructed formula $f(\gamma_{\mathbf{v}}, \gamma_{\mathbf{r}})$, the state q_0 is a distinguished initial state in $\mathfrak{M}^{\phi}$. Now, this is the case if the empty normative systems $\mathsf{N}_{\epsilon} \in \mathsf{WI}_{\mathcal{NE}}(\mathfrak{I}, q_0, f)$. □

Analogously to the previous result, we next investigate whether a given normative system strongly implements a normative behaviour function. In the following, we show that the verification version of the strong implementation problem is in $\mathbf{P}_{\parallel}^{\mathbf{\Sigma_2^P}[2]}$. This complexity class consists of all problems which can be solved by a polynomial-time deterministic oracle Turing machine which can make *two non-adaptive* queries to a problem in $\mathbf{\Sigma_2^P}$, cf. [67]. Non-adaptive means that queries must be independent from each other; in other words, it must be possible to perform them in parallel.

Theorem 5 *(Strong implementation: verification). Let $(\mathfrak{I}, q)$ be a pointed NIS, where $\mathfrak{I} = (\mathfrak{M}, \mathcal{P}refs, \mathcal{N})$ and $\mathsf{N} \in \mathcal{N}$ be a normative system. Let also f be a normative behaviour function. The problem whether $\mathsf{N} \in \mathsf{SI}_{\mathcal{NE}}(\mathfrak{I}, q, f)$ is in $\mathbf{P}_{\parallel}^{\mathbf{\Sigma_2^P}[2]}$. The problem is $\mathbf{\Sigma_2^P}$-hard as well as $\mathbf{\Pi_2^P}$-hard in the size of $\mathfrak{M}$, N and $\mathcal{P}refs$.*

Proof. *Membership:* We construct a deterministic polynomial time Turing machine (TM) N with an $\mathbf{\Sigma_2^P}$-oracle, implemented by a non-deterministic TM N' with **NP**-oracle. The TM N' works similar to the TM M of Theorem 4. N uses N' twice: to check whether $\mathcal{NE}(\Gamma(\mathfrak{M} \restriction \mathsf{N}, q, \vec{\gamma})) \neq \emptyset$ and whether for all $s \in \mathcal{NE}(\Gamma(\mathfrak{M} \restriction \mathsf{N}, q, \vec{\gamma}))$ it holds that $\pi(out_{\mathfrak{M} \restriction \mathsf{N}}(q,s)) \models^{\mathbf{LTL}} f(\vec{\gamma})$. The first part is done as in the proof of Theorem 4. For the second part, N calls N' with the additional input $\neg f(\vec{\gamma})$ and reverts the answer of N'. This shows that the problem is in $\mathbf{P}_{\parallel}^{\mathbf{\Sigma_2^P}[2]}$.

Sketch of hardness. The hardness results are proven in Theorem 21. The intuition is similar to the one given in Theorem 4 where the same two-player model $\mathfrak{M}^{\phi}$ is used, but the preference profiles of both players are changed. This case is technically more difficult as the verification problem consists of two parts: (i) ensuring that the set of Nash equilibria is non-empty; and (ii) ensuring that all Nash equilibria satisfy the outcome of the normative behavior function. For the $\mathbf{\Pi_2^P}$-hardness we also need to slightly modify the model $\mathfrak{M}^{\phi}$ by updating the roles of the verifier and the refuter. □

The next lemma shows that normative systems enjoy a small representation property: any update of a model by a normative systems can be obtained by a "small" normative system which is polynomial in the size of the model.

Lemma 6. *Let $\mathfrak{I} = (\mathfrak{M}, \mathcal{P}refs, \mathcal{N})$ be given where $\mathcal{N} \in \{\mathcal{N}_{\mathsf{r}}, \mathcal{N}_{\mathsf{s}}, \mathcal{N}_{\mathsf{rs}}\}$. For each $\mathsf{N} \in \mathcal{N}$ there is an $\mathsf{N}' \in \mathcal{N}$ with $|\mathsf{N}'| \leq 2 \cdot |Q| \cdot |Act^k|$ such that $\mathfrak{M} \restriction \mathsf{N} = \mathfrak{M} \restriction \mathsf{N}'$, where k is the number of agents in $\mathfrak{M}$.*

Proof. Let N be given, comprised of sanctioning norms SN and regimenting norms RN. For each $q \in Q$ and $\vec{\alpha} \in Act^k$ we define $S_{q,\vec{\alpha}} = \mathsf{San}_{\mathsf{N}}(\pi(q), \vec{\alpha})$ and $\pi^*(q)$ as the formula $\bigwedge_{\mathsf{p} \in \pi(q)} \mathsf{p} \wedge \bigwedge_{\mathsf{p} \notin \pi(q)} \neg \mathsf{p}$. We add the sanctioning norm $(\pi^*(q), \{\vec{\alpha}\}, S_{q,\vec{\alpha}})$ to N' if $\bot \notin S_{q,\vec{\alpha}}$, and $(\pi^*(q), \{\vec{\alpha}\}, \bot)$ otherwise. We observe that $|\mathsf{N}'| \leq |Q| \cdot |Act^k|$. It remains to show that $\mathfrak{M} \restriction \mathsf{N} = \mathfrak{M} \restriction \mathsf{N}'$. By Proposition 2.4 and the properties of N' we have that $\mathfrak{M} \restriction \mathsf{N}' = (\mathfrak{M} \restriction \mathsf{RN}') \restriction \mathsf{SN}'$. Thus, we can consider sanctioning and regimenting norms separately.

Firstly, we show that $\mathfrak{M} \restriction \mathsf{RN} = \mathfrak{M} \restriction \mathsf{RN}'$. Let q and $\vec{\alpha}$ be a state and action profile in $\mathfrak{M}$, respectively. It suffices to show that $\mathsf{App}_{\mathsf{RN}}(\pi(q), \vec{\alpha}) = \emptyset$ iff $\mathsf{App}_{\mathsf{RN}'}(\pi(q), \vec{\alpha}) = \emptyset$. "⇐": $\mathsf{App}_{\mathsf{RN}}(\pi(q), \vec{\alpha}) \neq \emptyset$ implies that there is an $\eta = (\varphi, \mathcal{A}, \bot) \in \mathsf{RN}$ with $\pi(q) \models \varphi$ and $\vec{\alpha} \in A$. This implies $(\pi^*(q), \{\vec{\alpha}\}, \bot) \in \mathsf{RN}'$ which implies $\mathsf{App}_{\mathsf{RN}'}(\pi(q), \vec{\alpha}) \neq \emptyset$. "⇒": $\mathsf{App}_{\mathsf{RN}'}(\pi(q), \vec{\alpha}) \neq \emptyset$

implies $\exists \eta = (\pi^*(q'), \{\vec{\alpha}\}, \bot) \in \mathsf{RN}'$ with $\pi(q) \models \pi^*(q')$. This implies $\exists \eta' = (\varphi, \mathcal{A}, \bot) \in \mathsf{RN}$ with $\vec{a} \in \mathcal{A}$ and $\pi(q') \models \varphi$. But then also $\pi(q) \models \varphi$ which implies $\mathsf{App}_{\mathsf{RN}}(\pi(q), \vec{\alpha}) \neq \emptyset$.

Secondly, we show that $\mathsf{San}_{\mathsf{SN}'}(\pi(q), \vec{\alpha}) = \mathsf{San}_{\mathsf{SN}}(\pi(q), \vec{\alpha})$ *for all* q *and* $\vec{\alpha}$ *in* $\mathfrak{M}$ *whenever* $\mathsf{App}_{\mathsf{RN}}(\pi(q), \vec{\alpha}) = \emptyset$:

$$\begin{aligned}
\mathsf{San}_{\mathsf{SN}'}(\pi(q), \vec{\alpha}) &= \bigcup \{S_{q',\vec{\alpha}} \mid (\pi^*(q'), \{\vec{\alpha}\}, S_{q',\vec{\alpha}}) \in \mathsf{App}_{\mathsf{SN}'}(\pi(q), \vec{\alpha})\} \\
&= \bigcup \{S_{q',\vec{\alpha}} \mid (\pi^*(q'), \{\vec{\alpha}\}, S_{q',\vec{\alpha}}) \in \mathsf{App}_{\mathsf{SN}'}(\pi(q), \vec{\alpha}), \pi(q) = \pi(q')\} \\
&= \bigcup \{S \mid (\varphi, \mathcal{A}, S) \in \mathsf{App}_{\mathsf{SN}}(\pi(q'), \vec{\alpha}), \pi(q) = \pi(q')\} \\
&= \bigcup \{S \mid (\varphi, \mathcal{A}, S) \in \mathsf{App}_{\mathsf{SN}}(\pi(q), \vec{\alpha})\} \\
&= \mathsf{San}_{\mathsf{SN}}(\pi(q), \vec{\alpha}).
\end{aligned}$$

This implies that the updates yield identical models. □

Now we consider the problem whether there is a normative systems that weakly implements a normative behaviour function.

Theorem 7 *(Weak implementation: existence). Let* $(\mathfrak{I}, q)$ *be a pointed NIS,* $\mathcal{N} \in \{\mathcal{N}_{\mathsf{rs}}, \mathcal{N}_{\mathsf{r}}, \mathcal{N}_{\mathsf{s}}\}$, $\mathsf{N} \in \mathcal{N}$*, and* f *be a normative behaviour function. The problem whether* $\mathsf{WI}_{\mathcal{NE}}(\mathfrak{I}, q, f) \neq \emptyset$ *is* $\mathbf{\Sigma_2^P}$*-complete.*

Proof. *Membership.* By Lemma 6, if $\mathsf{N} \in \mathsf{WI}_{\mathcal{NE}}(\mathfrak{I}, q, f)$ then there is also a "small" normative system $\mathsf{N}' \in \mathsf{WI}_{\mathcal{NE}}(\mathfrak{I}, q, f)$. We extend the algorithm described in the proof of Theorem 4 in such a way that the TM M guesses, in addition to the strategy profile, a "small" normative system N, and then works as before. Hardness follows from Theorem 4. □

For strong implementation it is no longer possible to guess a strategy profile but the normative behaviour function must be satisfied on all Nash equilibria. Thus, we can use the result of Theorem 5, but first a small normative system is guessed.

Theorem 8 *(Strong implementation: existence). Let* $(\mathfrak{I}, q)$ *be a pointed NIS and* $\mathcal{N} \in \{\mathcal{N}_{\mathsf{rs}}, \mathcal{N}_{\mathsf{s}}, \mathcal{N}_{\mathsf{r}}\}$, $\mathsf{N} \in \mathcal{N}$*, and* f *be a normative behaviour function. The problem whether* $\mathsf{SI}_{\mathcal{NE}}(\mathfrak{I}, q, f) \neq \emptyset$ *is* $\mathbf{\Sigma_3^P}$*-complete.*

Proof. *Membership.* By Lemma 6, if $\mathsf{N} \in \mathsf{SI}_{\mathcal{NE}}(\mathfrak{I}, q, f)$ then there is also a "small" normative system $\mathsf{N}' \in \mathsf{SI}_{\mathcal{NE}}(\mathfrak{I}, q, f)$. We construct a non-deterministic TM M which uses N' from Theorem 5 as oracle TM. M works as N from Theorem 5 but additionally guesses a small normative system N, and then works as N. This shows that the problem is in $\mathbf{NP}^{\mathbf{\Delta_3^P}} = \mathbf{\Sigma_3^P}$.

Sketch of hardness. The hardness proof for sanctioning norms and for regimentation norms is given in Theorems 25 and 29, respectively, where the basic intuition is similar to the one given in Theorems 4 and 5, the construction is slightly more sophisticated. As we now reduce $\mathbf{QSAT}_3$ to the strong implementation problem, we need to include the additional existential quantification in $\mathbf{QSAT}_3$ in the construction. For this purpose we encode a truth assignment of variables as a normative system. Guessing such a normative system corresponds to guessing a truth assignment of the newly existentially quantified variables. The difficulty in the construction is to ensure that the guessed normative system does not affect "good parts" of the model used in the construction of the two previous hardness results (Theorems 4 and 5) because after a normative system has been guessed the two players should essentially simulate the semantics of ϕ by guessing truth assignments of the variables under the scope of the other two quantifiers in the given $\mathbf{QSAT}_3$-formula, followed by playing the game theoretic game to evaluate the resulting propositional formula. □

Concluding remarks

In this section we analysed the complexity of the weak and strong implementation problem. The computational complexity of the membership problems, i.e. whether a given normative system weakly or strongly implements a normative behaviour function, respectively, were shown to be essentially located on the second level of the polynomial hierarchy [55]. In [37] it was shown that checking the existence of a pure Nash equilibrium is **NP**-complete: a strategy profile must be guessed and then verified, in polynomial time, whether it is a Nash equilibrium. The latter check *cannot* be done in polynomial time in the setting considered here, as the number of strategies of each player is exponential in the size of the model; there are $|Act|^{|Q|}$ many of these. Given this observation, a lower bound of $\mathbf{\Sigma_2^P} = \mathbf{NP}^{\mathbf{NP}}$ seems intuitive for weak implementation (an existential problem: *guess* a strategy profile with specific properties and verify it against *all* deviations) and a lower bound of $\mathbf{\Pi_2^P} = \mathbf{coNP}^{\mathbf{NP}}$ for strong implementation (a universal problem). Our results show that the complexities of the decision problems considered here are in line with these intuitive bounds: in Theorem 5 we show $\mathbf{\Sigma_2^P}$ as well as $\mathbf{\Pi_2^P}$ hardness and prove an $\mathbf{P}_{\parallel}^{\mathbf{\Sigma_2^P}[2]}$ upper bound. Moreover, we show that the weak implementation problem is no more difficult than the verification of checking that a given normative system weakly implements a normative behaviour function. The complexity of the strong implementation problem is located one level up in the polynomial hierarchy. The results do

also not appear that bad when compared with the complexity results of [69] in the context of Boolean games. The authors show that the problem whether there is an taxation scheme which ensures the existence of a pure Nash equilibrium with desirable properties is already $\mathbf{\Sigma_2^P}$-complete in the pure setting of Boolean games. Thus, we cannot hope for any better results for our weak implementation problem. That our strong implementation problem is more difficult than its weak version, $\mathbf{\Sigma_3^P}$-complete, stems from the fact that an additional normative system has to be guessed such that *all* Nash equilibria satisfy a specific property.

5. Multi-agent environment programs

The model of a multi-agent environment, as proposed in Section 2, is abstract in the sense that it assumes that the set of states and the state transitions are without an internal structure. In this section, we propose a specification language to program multi-agent environments. The introduction of this specification language allows us to program mechanisms and apply our formal methodology, as proposed in Section 3, to analyse the behaviour of such programs. In this way, one can check whether an environment program can ensure the system designer's objectives given that the participating agents behave according to their preferences. The proposed language allows the specification of an initial environment state as well as a set of synchronized actions. In environment programming, we are agnostic about individual agents and how they are programmed. We assume that a set of agents performing synchronized actions in the environment is given. These actions form the input of an environment program. The structural operational semantics of the language specifies the execution of programs. We show that the proposed language can be used to program a broad class of multi-agent environments as defined in Section 2. We extend the specification language with norms, as defined in Section 3, and present its operational semantics. We show that the extended language is expressive enough to program norm-governed multi-agent environments. In particular, we show that adding a set of norms to the program of a given multi-agent environment specifies the multi-agent environment updated with the set of norms.

In the rest of this section, we follow our abstract model of norm-based multi-agent systems and distinguish hard and soft facts. We assume that the states of norm-based environment programs is represented by hard and soft facts. We use $\Pi = \Pi_h \cup \Pi_s$ to denote the union of disjoint and finite sets of hard and soft facts (i.e. $\Pi_h \cap \Pi_s = \emptyset$), $\Pi_h^l = \{p, \neg p \mid p \in \Pi_h\}$ be the set of hard literals, $\Pi_s^l = \{p, \neg p \mid p \in \Pi_s\}$ be the set of soft literals, and $\Pi^l = \Pi_h^l \cup \Pi_s^l$.

5.1. Programming multi-agent environments

A multi-agent environment can be programmed by specifying its initial state and a set of action profiles. The initial state of an environment program is specified by a set of atomic propositions and the action profiles are specified in terms of pre- and postconditions. The precondition of an action profile is a set of literals that specify the states of the environment programs in which the performance of the action profile results in a state transition. The resulting state is determined by adding the positive atoms of the action's postcondition to the state in which the action is performed and removing the negative atoms of the postcondition from it. The pre- and postconditions of action profiles are assumed to consist of hard facts such that action profiles are activated by hard facts and change only those facts of the program states. The performance of an action profile by individual agents in a state that does not satisfy its precondition is assumed to have no effect on the state and considered as looping in that state. Please note that in environment programming we are agnostic about individual agents and their internals. We assume that the agents decide and perform synchronized actions, and that the environment program realises the effect of the actions according to their specified pre- and postconditions. So, it is possible that the performance of actions by individual agents do not change the environment state. This is also the case with the performance of any action profile that is not specified by the environment program.

Definition 15 *(Multi-agent environment program).* Let $\mathbb{A}\text{gt} = \{1, \ldots, k\}$ be a set of agents. A multi-agent environment program (over a finite set of propositional symbols Π as introduced above) is a tuple $(F_0, (Act_1, \ldots, Act_k), \mathcal{A}_{spec})$, where

- $F_0 \subseteq \Pi$ is the initial state of the environment program,
- Act_i is the set of actions of agent $i \in \mathbb{A}\text{gt}$,
- $\mathcal{A}_{spec} \subseteq \{\ (Pre, \vec{\alpha}, Post) \ \mid\ \vec{\alpha} \in Act_1 \times \ldots \times Act_k \ \ and \ \ Pre, Post \subseteq \Pi_h^l\}$ is a subset of action profile specifications, where we assume that each action tuple $\vec{\alpha}$ can be included in at most one action profile specification.

We assume that k agents operate in a multi-agent environment such that $\vec{\alpha}$ is an action profile of the form $(\alpha_1, \ldots, \alpha_k)$, where $\alpha_i \in Act_i$ is the name of the action performed by agent $1 \leq i \leq k$. It is important to note that pre- and postconditions are not assigned to the actions of individual agents, but to action profiles. This implies that some possible action profiles from $Act_1 \times \ldots \times Act_k$ may not be specified in $\mathcal{A}_{spec}$ in which case we assume that their executions will not change the state of the multi-agent environment program and loop in those states. Finally, we follow the convention, similar to the programming language Prolog [13], to use "_" as a place-holder for any action in an action profile specification.[12] For

[12] Thus, _ plays the same role as $\star$ in the context of CGS. We use Prolog's notation here to emphasize that we are working in a program setting.

example, we use $(Pre, (M, \_), Post)$ to indicate that the performance of action profiles (M, M) and (M, W) in states that satisfy the precondition Pre results in states that satisfies the postcondition $Post$.

Example 16 *(Environment program for the narrowed road).* Let $\mathbb{A}\text{gt} = \{1, 2\}$, $Act_i = \{M, W\}$ for $i \in \{1, 2\}$ be the actions that can be performed by the cars in our running example, and propositional symbol p_i^X be interpreted as before. The running example can be implemented by the multi-agent environment program $(\{p_1^s, p_2^s\}, (Act_1, Act_2), \{a_1, \ldots, a_5\})$, where

$$
\begin{array}{lllll}
a_1 = & (\ \{p_1^s, p_2^s\}, & (W, M), & \{p_2^e, \neg p_2^s\} &) \\
a_2 = & (\ \{p_1^s, p_2^s\}, & (M, W), & \{p_1^e, \neg p_1^s\} &) \\
a_3 = & (\ \{p_1^s, p_2^s\}, & (M, M), & \{p_1^m, p_2^m, \neg p_1^s, \neg p_1^s\} &) \\
a_4 = & (\ \{p_1^s, p_2^e\}, & (M, \_), & \{p_1^e, \neg p_1^s\} &) \\
a_5 = & (\ \{p_1^e, p_2^s\}, & (\_, M), & \{p_2^e, \neg p_2^s\} &)
\end{array}
$$

Note that a_i is used to denote an action profile specification, while $\vec{\alpha}_i$ denotes an action profile. Starting from the initial state of a multi-agent environment program, an execution of the program changes the program state depending on the agents' actions (the input of the environment program). An arbitrary state of an environment program is specified by a set of atomic propositions $F \subseteq \Pi$. The structural operational semantics of the multi-agent environment programs are specified by a set of transition rules, each specifies how the environment program state changes when agents perform actions. Therefore, in the sequel, we use $(F, (Act_1, \ldots, Act_k), \mathcal{A}_{spec})$ to denote an environment program in state F. Since the agents' action repertoire and the action specifications do not change during the execution of the program, we only use the state of the environment program F (instead of $(F, (Act_1, \ldots, Act_k), \mathcal{A}_{spec})$) in the transition rules. We note that not every state F is necessarily reachable from the initial state.

Definition 16 *(Structural operational semantics).* Let $(F, (Act_1, \ldots, Act_k), \mathcal{A}_{spec})$ be an environment program in state $F \subseteq \Pi$ and $\vec{\alpha} \in Act_1 \times \ldots \times Act_k$. For $Pre, Post \subseteq \Pi_h^l$ we write $F \vdash Pre$ *iff* $Pre \cap \Pi \subseteq F$ and $(Pre \setminus \Pi) \cap F = \emptyset$ (the precondition Pre is derivable from the facts F iff all positive atoms in Pre are in F and all negative atoms in Pre are not in F). Further, we define $F \oplus Post = (F \setminus \{p \mid \neg p \in Post\}) \cup \{p \mid p \in Post\}$ (updating F with postcondition $Post$ removes negative atoms in $Post$ from F and adds positive atoms in $Post$ to F). The following three transition rules specify possible execution steps of the environment program in state F.

$$
\frac{(Pre, \vec{\alpha}, Post) \in \mathcal{A}_{spec} \text{ and } F \vdash Pre \text{ and } F' = F \oplus Post}{F \xrightarrow{\vec{\alpha}} F'}
$$

$$
\frac{(Pre, \vec{\alpha}, Post) \in \mathcal{A}_{spec} \text{ and } F \nvdash Pre}{F \xrightarrow{\vec{\alpha}} F}
$$

$$
\frac{\forall Pre, Post \subseteq \Pi_h^l : (Pre, \vec{\alpha}, Post) \notin \mathcal{A}_{spec}}{F \xrightarrow{\vec{\alpha}} F}
$$

The first transition rule specifies the execution step of the environment program based on the action profile $\vec{\alpha}$ when its precondition is satisfied. Such an execution step updates the set of facts F with the postcondition of the action profile $\vec{\alpha}$. The second transition rule is the same except that it applies when the precondition of $\vec{\alpha}$ is not satisfied. Such an execution step does not change the state of the environment program. Finally, the third rule specifies the execution step of the environment program based on an unspecified action profile $\vec{\alpha}$. Such an execution step does not change the state of the environment program.

In the following, we use $\mathcal{T}_{\text{basic}}$ to denote the set of transition rules from Definition 16, $trans(F, Act_1, \ldots, Act_k, \mathcal{A}_{spec}, \mathcal{T}_{\text{basic}})$ to denote the set of all transitions that are derivable from transition rules $\mathcal{T}_{\text{basic}}$ based on the environment program $(F, (Act_1, \ldots, Act_k), \mathcal{A}_{spec})$, and $trans(Act_1, \ldots, Act_k, \mathcal{A}_{spec}, \mathcal{T}_{\text{basic}})$ to denote the set of all transitions that are derivable from transition rules $\mathcal{T}_{\text{basic}}$ based on environment programs $(F, (Act_1, \ldots, Act_k), \mathcal{A}_{spec})$ for any $F \subseteq \Pi$. The latter is the set of all possible transitions based on transition rules $\mathcal{T}_{\text{basic}}$, actions $Act_1, \ldots, Act_k$, action specifications $\mathcal{A}_{spec}$, and all sets of facts $F \subseteq \Pi$.

An execution of an environment program consists of subsequent execution steps resulting in a sequence of program states. In order to define the set of all possible executions of an environment program, we first define the set of all possible states that can be generated (reached) from an arbitrary state by applying subsequent transitions.

Definition 17 *(Program states).* Let $F \subseteq \Pi$ be a set of facts and τ be a set of transitions. The set of states reached from F by subsequent transitions from τ, denoted by $\mathsf{gen}(\tau, F)$, is defined as follows:

$$
\mathsf{gen}(\tau, F) := \{F\} \cup \bigcup_{i=1}^{\infty} \mathsf{suc}_\tau^i(\{F\}) \quad \text{where}
$$

$$\mathsf{suc}^i_\tau(X) = \underbrace{\mathsf{suc}_\tau(\ldots(\mathsf{suc}_\tau}_{i \text{ times}}(X))\ldots) \quad \text{where } X \subseteq \mathcal{P}(\Pi) \text{ and}$$

$$\mathsf{suc}_\tau(X) = \{F_2 \mid F_1 \xrightarrow{\vec{\alpha}} F_2 \in \tau \text{ and } F_1 \in X\}.$$

Observe that $\mathsf{gen}(\tau, F)$ is finite when τ is finite. Given an environment program and the set of transition rules $\mathcal{T}_{\text{basic}}$, the set of possible executions of the environment program generates a concurrent game structure (as specified in Section 2).

Definition 18 *(Programs $\rightsquigarrow$ CGS).* Let $(F_0, (Act_1, \ldots, Act_k), \mathcal{A}_{spec})$ be an environment program, $\mathcal{T}_{\text{basic}}$ be the set of transition rules as defined in Definition 16 and $\tau_b = trans(Act_1, \ldots, Act_k, \mathcal{A}_{spec}, \mathcal{T}_{\text{basic}})$. The environment program $\mathcal{T}_{\text{basic}}$-generates a pointed CGS $(\mathfrak{M}, F_0)$ with $\mathfrak{M} = (\mathbb{A}\text{gt}, Q, \Pi, \pi, Act, d, o)$ as follows:

- $\mathbb{A}\text{gt} = \{1, \ldots, k\}$
- $Q = \mathsf{gen}(\tau_b, F_0)$
- $\Pi = F_0 \cup \{p \mid (Pre, \vec{\alpha}, Post) \in \mathcal{A}_{spec} \text{ and } p \in Post\}$
- $\pi(F) = F$ for $F \in Q$
- $Act = Act_1 \cup \ldots \cup Act_k$
- $d_i(F) = Act_i$ for $i \in \mathbb{A}\text{gt}$ and $F \in Q$
- $o(F, \vec{\alpha}) = F'$ for $F, F' \in Q$, $\vec{\alpha} \in d_1(F) \times \ldots \times d_k(F)$, and $F \xrightarrow{\vec{\alpha}} F' \in \tau_b$

The translation between environment programs and concurrent game structures is restricted to specific classes of concurrent game structures as specified in the next definition. In the following, we use also variables $q_0, q_1, \ldots$ to range over the set of states Q.

Definition 19 *(Finite, distinguished, generated CGS).* Let $\mathfrak{M} = (\mathbb{A}\text{gt}, Q, \Pi, \pi, Act, d, o)$ be a concurrent game structure. We introduce the following notation:

- $\mathfrak{M}$ is called finite if Q is finite,
- $\mathfrak{M}$ is called distinguished if all states differ in their valuations, i.e., for all $q, q' \in Q$ with $q \neq q'$ we have that $\pi(q) \neq \pi(q')$, and
- $\mathfrak{M}$ is called uniform if for all $i \in \mathbb{A}\text{gt}$ and all $q, q' \in Q$ we have that $d(i, q) = d(i, q')$, i.e., the agents have the same options in every state.

The following proposition formulates the relation between environment programs and their corresponding CGSs.

Proposition 9. *Let Π be the set of propositional symbols, $(F_0, (Act_1, \ldots, Act_k), \mathcal{A}_{spec})$ be an environment program and $(\mathfrak{M}, F_0)$ be the pointed CGS that is $\mathcal{T}_{\text{basic}}$-generated by the environment program. If Π is finite, then $\mathfrak{M}$ is finite. Moreover, $\mathfrak{M}$ is distinguished and uniform.*

Proof. The sets of atoms in $\mathsf{gen}(trans(Act_1, \ldots, Act_k, \mathcal{A}_{spec}, \mathcal{T}_{\text{basic}}), F_0)$, which determine the set of states in Q, are subsets of Π such there can be only a finite number of them as Π is finite. Moreover, that $\mathfrak{M}$ is distinguished follows directly from the fact that all sets in $\mathsf{gen}(trans(Act_1, \ldots, Act_k, \mathcal{A}_{spec}, \mathcal{T}_{\text{basic}}), F_0)$ are different. Finally, that $\mathfrak{M}$ is uniform follows directly from the fact that for each $i \in \mathbb{A}\text{gt}$ it holds: $d(i, q) = Act_i$ for all states $q \in Q$, i.e., each agent has one and the same set of options in every state. □

5.2. Multi-agent environment programs with norms

The environment programs as defined in the previous subsection do not account for norms and norm enforcement. In order to add norms to multi-agent environment programs and to enforce them during program executions, we use (sanctioning and regimenting) norms as introduced in Definition 9. A norm is thus represented as a triple $(\varphi, \mathcal{A}, S)$, where φ is a propositional formula, $\mathcal{A}$ is a set of action profiles, and either $S \subseteq \Pi_s$ (sanctioning norm) or $S = \bot$ (regimenting norm). In the rest of this paper, we just use the term norm when the distinction between sanctioning and regimenting norms is not relevant. Like before, the pre- and postconditions of action profiles are assumed to consist of hard facts only such that adding norms does not change the activation and effect of action profiles. As explained before, throughout this section we assume that $\Pi = \Pi_h \cup \Pi_s$ is a finite set of propositional symbols.

Definition 20 *(Norm-based multi-agent environment program).* Let $\mathbb{A}\text{gt} = \{1, \ldots, k\}$ be a set of agents. A norm-based multi-agent environment program is a tuple $(F_0, (Act_1, \ldots Act_k), \mathcal{A}_{spec}, \mathsf{N})$, where

- $F_0 \subseteq \Pi$,
- Act_i and $\mathcal{A}_{spec}$ are as introduced in Definition 15, and
- N is a set of (sanctioning and regimenting) norms as introduced in Definition 9.

Following the narrowed road environment program as specified in Example 16, the behaviour of cars can be regulated by adding norms to the environment program.

Example 17 *(Norm-based environment program for the narrowed road).* Let $(\{p_1^s, p_2^s\}, (Act_1, Act_2), \{a_1, \ldots, a_5\})$ be an environment program as specified in Example 16, and $\mathsf{N} = \{(p_1^s \wedge p_2^s, \{(W, W)\}, \{\mathsf{fine}_1\}),\ (p_1^s \wedge p_2^s, \{(\_, M)\}, \{\mathsf{fine}_2\})\}$ be a set of norms as explained in Example 10. The norm-based environment program $(\{p_1^s, p_2^s\}, (Act_1, Act_2), \{a_1, \ldots, a_5\}, \mathsf{N})$ implements the narrowed road example where norms N are enforced.

The executions of norm-based multi-agent environment programs are similar to the executions of multi-agent environment programs (without norms) in the sense that both update the program states with the effects of action profiles. However, the execution steps of a norm-based multi-agent environment program proceed by enforcing norms to the resulting program states. The enforcement of norms on a program state consists of updating the state with the consequences of applicable norms. So, in order to define the execution steps of norm-based multi-agent environment programs, we need to determine the norms that are applicable in a program state and their consequences. For this purpose, we use the function $\mathsf{San}_{\mathsf{N}}(X, \vec{\alpha})$, as defined in Definition 10, to determine the sanction set that should be imposed when the agents perform action profile $\vec{\alpha}$ in state X. Note that the sanction set may be $\{\bot\}$, which means that a regimenting norm should be enforced. In this case, we have to ensure that the performance of $\vec{\alpha}$ does not have any effect on state X.

We distinguish two general cases depending on whether the sanction set is $\{\bot\}$ or not. When applicable norms are sanctioning norms (i.e., resulting in a set of soft facts), we update the resulting program state with the sanctions. We define the update of a program state with sanctions as being non-cumulative in the sense that previous sanctions are removed before new sanctions are added. We use the update function $\otimes$ to update a state with sanctions. Note that in Definition 21 $\otimes$ removes all sanctions (soft facts) before adding new ones. This ensures that sanctions become non-cumulative in transitions. This update function should not be confused with $\oplus$, which is used to update program states with the postcondition of action profiles.

The effect of an action profile on an environment program state in the context of some existing norms is specified by distinguishing four cases: 1) the action specification is given and its precondition is satisfied, 2) the action specification is given but its precondition is not satisfied, 3) the action specification is not given, and 4) the action profile is regimented by some norms. In the first case, we update the program state with the postcondition of the action profile and then with possible sanctions, while in the second and third case the program state is updated only with possible sanctions. These two cases indicate that the performance of an action profile in a program state will be sanctioned (if there is a sanctioning norms applicable) even when it has a specification but its precondition is not satisfied or when the action profile has no specification. These two cases capture the intuition that any unsuccessful attempt to violate norms will be sanctioned as well. Note that the sanction set in the first three cases can be an empty set if no norm is applicable. Finally, in the fourth case, when an action is regimented by some norms, the action will have no effect on the environment state.

Definition 21 *(Structural operational semantics).* Let $(F, (Act_1, \ldots, Act_k), \mathcal{A}_{spec}, \mathsf{N})$ be a norm-based environment program in an arbitrary state $F \subseteq \Pi$, $\vec{\alpha} \in Act_1 \times \ldots \times Act_k$, and N be a set of norms as defined in Definition 9. Let also $\oplus$ and $\vdash$ be defined as in Definition 16. Finally, let $F \otimes S = (F \setminus \Pi_s) \cup S$ for $S \subseteq \Pi_s$. The following four transition rules specify possible execution steps of the norm-based environment program.

$$\frac{(Pre, \vec{\alpha}, Post) \in \mathcal{A}_{spec} \text{ and } F \vdash Pre \text{ and } F' = (F \oplus Post) \otimes \mathsf{San}_{\mathsf{N}}(F, \vec{\alpha}) \text{ and } \mathsf{San}_{\mathsf{N}}(F, \vec{\alpha}) \neq \{\bot\}}{F \xrightarrow{\vec{\alpha}} F'}$$

$$\frac{(Pre, \vec{\alpha}, Post) \in \mathcal{A}_{spec} \text{ and } F \nvdash Pre \text{ and } F' = F \otimes \mathsf{San}_{\mathsf{N}}(F, \vec{\alpha}) \text{ and } \mathsf{San}_{\mathsf{N}}(F, \vec{\alpha}) \neq \{\bot\}}{F \xrightarrow{\vec{\alpha}} F'}$$

$$\frac{\forall Pre, Post \subseteq \Pi_h^l : (Pre, \vec{\alpha}, Post) \notin \mathcal{A}_{spec} \text{ and } F' = F \otimes \mathsf{San}_{\mathsf{N}}(F, \vec{\alpha}) \text{ and } \mathsf{San}_{\mathsf{N}}(F, \vec{\alpha}) \neq \{\bot\}}{F \xrightarrow{\vec{\alpha}} F'}$$

$$\frac{\mathsf{San}_{\mathsf{N}}(F, \vec{\alpha}) = \{\bot\}}{F \xrightarrow{\vec{\alpha}} F}$$

The first transition rule applies when action profile $\vec{\alpha}$ is performed, its precondition holds, and no regimenting norms are triggered. The second transition rule is the same except that the precondition of $\vec{\alpha}$ is not satisfied. In this case applicable sanctioning norms are enforced without realising the effect (i.e., postcondition) of $\vec{\alpha}$. The third transition rule applies when an unspecified action profile $\vec{\alpha}$ is performed and no regimenting norms are triggered. In this case, applicable sanctioning norms are enforced. Note that the environment is assumed to be exogenous to agents in the sense that agents decide on

which actions to perform independently of the environment specification. This allows agents to decide and perform actions for which there is no environment specification. It is also important to note that a norm specifies that the performance of an action profile in a state should be sanctioned, regardless of the specification of the action profile. Thus, agents can perform an unspecified action profile for which a sanction may occur. The first three transition rules ensure that sanctioning norms are enforced regardless of the specification of action profiles. These transition rules capture the idea that any (successful or unsuccessful) attempt to violate norms is sanctioned. Note also that in the first three transition rules no sanctioning norm needs to be applicable in state X, i.e., $\mathsf{San}_{\mathsf{N}}(X, \vec{\alpha})$ can be an empty set indicating that action profile α does not violate any norm in state X. Finally, the fourth transition rule applies when the performance of an action profile triggers regimenting norms, again regardless of the specification of action profiles. It is important to notice that sanctions do not accumulate through consecutive states. The following proposition shows that the sanctions imposed on a state (propositional symbols from Π_s) are only those caused by the action performed in the previous state.

In the following, we use $\mathcal{T}_{\text{norm}}$ for the set of transition rules from Definition 21, $\mathit{trans}(F, \mathit{Act}_1, \ldots, \mathit{Act}_k, \mathcal{A}_{\mathit{spec}}, \mathsf{N}, \mathcal{T}_{\text{norm}})$ to denote the set of all transitions that are derivable from transition rules $\mathcal{T}_{\text{norm}}$ based on the norm-based environment program $(F, (\mathit{Act}_1, \ldots, \mathit{Act}_k), \mathcal{A}_{\mathit{spec}}, \mathsf{N})$, and $\mathit{trans}(\mathit{Act}_1, \ldots, \mathit{Act}_k, \mathcal{A}_{\mathit{spec}}, \mathsf{N}, \mathcal{T}_{\text{norm}})$ to denote the set of all transitions that are derivable from transition rules $\mathcal{T}_{\text{norm}}$ based on norm-based environment programs $(F, (\mathit{Act}_1, \ldots, \mathit{Act}_k), \mathcal{A}_{\mathit{spec}}, \mathsf{N})$ for all $F \subseteq \Pi$.

Proposition 10. *Let $(F, (\mathit{Act}_1, \ldots, \mathit{Act}_k), \mathcal{A}_{\mathit{spec}}, \mathsf{N})$ be a norm-based environment program in an arbitrary program state F, $F \xrightarrow{\vec{\alpha}} F' \in \mathit{trans}(\mathit{Act}_1, \ldots, \mathit{Act}_k, \mathcal{A}_{\mathit{spec}}, \mathsf{N}, \mathcal{T}_{\mathit{norm}})$ and $S = \mathsf{San}_{\mathsf{N}}(F, \vec{\alpha})$. Then, we have*

$$F' \cap \Pi_s = \begin{cases} S & \text{if } S \neq \{\bot\} \\ F \cap \Pi_s & \text{otherwise.} \end{cases}$$

Proof. Directly follows from the definition of $\otimes$ and the transition rules in Definition 21. □

For a given norm-based environment program, the set of transition rules $\mathcal{T}_{\text{norm}}$ generates a concurrent game structure. We use Definition 18 to define the concurrent game structure, which is said to be generated by the norm-based environment program. Note that Definition 18 can be applied as norm-based environment programs have the required ingredients such as an initial state F_0, a sets of actions for each agent, and a set of action profile specifications. There is, however, one difference between environment programs and norm-based environment programs which requires a slight modification of Definition 18. The difference is that we now assume the set of transitions is $\mathit{trans}(\mathit{Act}_1, \ldots, \mathit{Act}_k, \mathcal{A}_{\mathit{spec}}, \mathsf{N}, \mathcal{T}_{\text{norm}})$, instead of $\mathit{trans}(\mathit{Act}_1, \ldots, \mathit{Act}_k, \mathcal{A}_{\mathit{spec}}, \mathcal{T}_{\text{basic}})$. This means that we use $\mathsf{gen}(\mathit{trans}(\mathit{Act}_1, \ldots, \mathit{Act}_k, \mathcal{A}_{\mathit{spec}}, \mathsf{N}, \mathcal{T}_{\text{norm}}), F_0)$ to generate the set of reachable states, and $F \xrightarrow{\vec{\alpha}} F' \in \mathit{trans}(\mathit{Act}_1, \ldots, \mathit{Act}_k, \mathcal{A}_{\mathit{spec}}, \mathsf{N}, \mathcal{T}_{\text{norm}})$ to indicate that the transition $F \xrightarrow{\vec{\alpha}} F'$ is derivable based on $\mathcal{T}_{\text{norm}}$. The formal definitions are as before.

Definition 22 *(Norm-based programs $\rightsquigarrow$ CGS).* Let $(F_0, (\mathit{Act}_1, \ldots, \mathit{Act}_k), \mathcal{A}_{\mathit{spec}}, \mathsf{N})$ be a norm-based environment program and $\tau_n = \mathit{trans}(\mathit{Act}_1, \ldots, \mathit{Act}_k, \mathcal{A}_{\mathit{spec}}, \mathsf{N}, \mathcal{T}_{\text{norm}})$. The program is said to $\mathcal{T}_{\text{norm}}$-generate a pointed CGS $(\mathfrak{M}, F_0)$, where $\mathfrak{M} = (\mathbb{A}\text{gt}, Q, \Pi, \pi, \mathit{Act}, d, o)$ is defined as follows:

- $\mathbb{A}\text{gt}, \Pi, \pi, \mathit{Act}, d$ are specified as in Definition 18,
- $Q = \mathsf{gen}(\tau_n, F_0)$, and
- $o(F, \vec{\alpha}) = F'$ for $F, F' \in Q, \vec{\alpha} \in d_1(F) \times \ldots \times d_k(F)$ and $F \xrightarrow{\vec{\alpha}} F' \in \tau_n$.

The class of concurrent game structures generated by norm-based environment programs is characterized by the following proposition.

Proposition 11. *Let Π be the set of propositional symbols, $(F_0, (\mathit{Act}_1, \ldots, \mathit{Act}_k), \mathcal{A}_{\mathit{spec}}, \mathsf{N})$ be a norm-based environment program, and $(\mathfrak{M}, F_0)$ be the pointed CGS that is $\mathcal{T}_{\mathit{norm}}$-generated by it. If Π is finite, then $\mathfrak{M}$ is finite. Moreover, $\mathfrak{M}$ is distinguished and uniform.*

Proof. Similar to the proof of Proposition 9. Observe that the elements of $Q = \mathsf{gen}(\mathit{trans}(\mathit{Act}_1, \ldots, \mathit{Act}_k, \mathcal{A}_{\mathit{spec}}, \mathsf{N}, \mathcal{T}_{\text{norm}}), F_0)$ are subsets of Π and there are only finitely many of them when Π is finite. Observe also that $\mathfrak{M}$ is distinguished as the elements of Q are distinct and that $\mathfrak{M}$ is uniform as $d_i(q) = \mathit{Act}_i$ for all $i \in \mathbb{A}\text{gt}$ and $q \in Q$. □

The generated CGS does not accumulate sanctions through consecutive states.

Proposition 12. *Let $(F_0, (\mathit{Act}_1, \ldots, \mathit{Act}_k), \mathcal{A}_{\mathit{spec}}, \mathsf{N})$ be a norm-based environment program that $\mathcal{T}_{\mathit{norm}}$-generates $(\mathfrak{M}, F_0)$, where $\mathfrak{M} = (\mathbb{A}\text{gt}, Q, \Pi, \pi, \mathit{Act}, d, o)$. For $F \in Q, \vec{\alpha} \in \mathit{Act}_1 \times \ldots \times \mathit{Act}_k$ and $S = \mathsf{San}_{\mathsf{N}}(F, \vec{\alpha})$, we have*

$$\pi(o(F,\vec{\alpha})) \cap \Pi_s = \begin{cases} S & \text{if } S \neq \{\bot\} \\ \pi(F) \cap \Pi_s & \text{otherwise.} \end{cases}$$

Proof. From Definition 22 we have $o(F,\vec{\alpha}) = F'$ iff $F \xrightarrow{\vec{\alpha}} F' \in trans(Act_1,\ldots,Act_k,\mathcal{A}_{spec},\mathsf{N},\mathcal{T}_{\text{norm}})$ and Proposition 10 we know that sanctions are not accumulated through transitions. □

Note the correspondence between Proposition 12 and Proposition 1 (on page 109). The execution of a norm-based environment program may generate a different set of states than the execution of the corresponding environment program without norms does. This is due to the fact that the performance of unspecified action profiles or the performance of specified action profiles for which the precondition does not hold can now be sanctioned resulting in new states. Note that the performance of such an action profile in an environment program without norms results in the same state. Also, the application of regimenting norms may prevents reaching new states. These observations are formulated in the following proposition.

Proposition 13. *Let* $(F_0,(Act_1,\ldots,Act_k),\mathcal{A}_{spec})$ *be an environment program,* $(F_0,(Act_1,\ldots,Act_k),\mathcal{A}_{spec},\mathsf{RN}\cup\mathsf{SN})$ *be the environment program with regimenting norms* RN *and sanctioning norms* SN*. Let also* $\tau_b = trans(Act_1,\ldots,Act_k,\mathcal{A}_{spec},\mathcal{T}_{basic})$ *and* $\tau_n = trans(Act_1,\ldots,Act_k,\mathcal{A}_{spec},\mathsf{RN}\cup\mathsf{SN},\mathcal{T}_{norm})$*. Then, we have:*

- *When the sets of regimentation and sanctioning norms are empty, i.e.,* $\mathsf{RN}\cup\mathsf{SN}=\emptyset$*, we have* $\mathsf{gen}(\tau_n,F_0)=\mathsf{gen}(\tau_b,F_0)$*.*
- *When there are only regimentation norms and no sanctioning norms, i.e.,* $\mathsf{RN}\neq\emptyset$ *and* $\mathsf{SN}=\emptyset$*, we have* $\mathsf{gen}(\tau_n,F_0)\subseteq\mathsf{gen}(\tau_b,F_0)$*.*
- *When there are only sanctioning norms and no regimentation norms, i.e.,* $\mathsf{RN}=\emptyset$ *and* $\mathsf{SN}\neq\emptyset$*, we have* $|\mathsf{gen}(\tau_n,F_0)|\geq|\mathsf{gen}(\tau_b,F_0))|$*.*

Proof.

- Since $\mathsf{RN}\cup\mathsf{SN}=\emptyset$, we have for all $F\subseteq\Pi$ and all $\vec{\alpha}\in Act_1\times\ldots\times Act_k$: $\mathsf{San}_{\mathsf{N}}(F,\vec{\alpha})=\emptyset$. This makes the transition rules $\mathcal{T}_{\text{basic}}$ and $\mathcal{T}_{\text{norm}}$ identical.
- Let $\mathsf{RN}\neq\emptyset$ and $\mathsf{SN}=\emptyset$. Following Definition 17, we have that $gen(\tau,F) := \{F\}\cup\bigcup_{i=1}^{\infty}\mathsf{suc}^i_\tau(\{F\})$. We show that if $F\in\{F_0\}\cup\bigcup_{i=1}^{l}\mathsf{suc}^i_{\tau_n}(\{F_0\})$ then also $F\in\{F_0\}\cup\bigcup_{i=1}^{l}\mathsf{suc}^i_{\tau_b}(\{F_0\})$ by induction on l. The base case, $l=1$, is trivial. Suppose the claim holds for $l>1$. Let $F\in\{F_0\}\cup\bigcup_{i=1}^{l+1}\mathsf{suc}^i_{\tau_n}(\{F_0\})$. Then, there must be an $F'\in\{F_0\}\cup\bigcup_{i=1}^{l}\mathsf{suc}^i_{\tau_n}(\{F_0\})$ with $F'\xrightarrow{\vec{\alpha}}F\in\tau_n$. If the latter transition is obtained by a regimentation of $\vec{\alpha}$ in F' then $F=F'$ and by induction hypothesis we obtain that $F\in\{F_0\}\cup\bigcup_{i=1}^{l+1}\mathsf{suc}^i_{\tau_b}(\{F_0\})$. If the action is not regimented, then we also have that $F'\xrightarrow{\vec{\alpha}}F\in\tau_b$ and thus $F\in\mathsf{suc}_{\tau_b}(\{F'\})$. Then, as by induction hypothesis $F'\in\{F_0\}\cup\bigcup_{i=1}^{l}\mathsf{suc}^i_{\tau_b}(\{F_0\})$, we also obtain that $F\in\{F_0\}\cup\bigcup_{i=1}^{l+1}\mathsf{suc}^i_{\tau_b}(\{F_0\})$. The claim follows.
- Let $F\subseteq\Pi$, $F\xrightarrow{\vec{\alpha}}F\in\tau_b$ for some $\vec{\alpha}\in Act_1\times\ldots\times Act_k$, and $\mathsf{San}_{\mathsf{SN}}(F,\vec{\alpha})\neq\emptyset$ (note that $\bot\notin\mathsf{San}_{\mathsf{SN}}(F,\vec{\alpha})$). Then, we have $F\xrightarrow{\vec{\alpha}}F\oplus\mathsf{San}_{\mathsf{SN}}(F,\vec{\alpha})\in\tau_n$. This implies that $F\oplus\mathsf{San}_{\mathsf{SN}}(F,\vec{\alpha})\in\mathsf{gen}(\tau_n,F_0)$ but $F\oplus\mathsf{San}_{\mathsf{SN}}(F,\vec{\alpha})\notin\mathsf{gen}(\tau_b,F_0)$. □

5.3. Relation between norm-based update and norm-based environment

We now investigate the relation between the concurrent game structure $\mathfrak{M}^n$ that is generated by a norm-based environment program, and the concurrent game structure $\mathfrak{M}'$ that is generated by the corresponding environment program without norms (i.e., the same initial state, actions, and action specifications) which is then updated with the same norms. We first show that if states q^n and q' from $\mathfrak{M}^n$ and $\mathfrak{M}'$, respectively, have the same valuation, then the states reached from q^n and q' by the same action profile have the same valuation as well. This is formulated by the following lemma.

Lemma 14. *Let* $(F_0,(Act_1,\ldots,Act_k),\mathcal{A}_{spec})$ *be an environment program that* $\mathcal{T}_{basic}$*-generates the concurrent game structure* $(\mathfrak{M},F_0)$ *where* $\mathfrak{M}=(\mathbb{A}\text{gt},Q,\Pi,\pi,Act,d,o)$*. Let* $(F_0,(Act_1,\ldots,Act_k),\mathcal{A}_{spec},\mathsf{N})$ *be a norm-based environment program that* $\mathcal{T}_{norm}$*-generates the concurrent game structure* $(\mathfrak{M}^n,F_0)$ *where* $\mathfrak{M}^n=(\mathbb{A}\text{gt},Q^n,\Pi,\pi^n,Act,d^n,o^n)$*. Let* $\mathfrak{M}'=\mathfrak{M}\upharpoonright\mathsf{N}$*, where* $\mathfrak{M}'=(\mathbb{A}\text{gt},Q',\Pi,\pi',Act,d',o')$*. For any* $F^n\in Q^n$*,* $F'\in Q'$ *and* $\vec{\alpha}\in Act_1\times\ldots\times Act_k$*, we have that if* $\pi^n(F^n)=\pi'(F')$ *then* $\pi^n(o^n(F^n,\vec{\alpha}))=\pi'(o'(F',\vec{\alpha}))$*.*

Proof. First, we note that by Proposition 11 we have that $\mathfrak{M}$ and $\mathfrak{M}^n$ are uniform. Therefore, by the definition of norm update given in Definition 11 model $\mathfrak{M}'$ is uniform as well. As a consequence, in all states of all three models all action tuples from $Act_1\times\ldots\times Act_k$ are available. Let $F^n\in Q^n$, $F'\in Q'$ with $\pi^n(F^n)=\pi'(F')$. We note that $F',F^n\subseteq\Pi$, or $F^n\subseteq\Pi$ and $F'=(F,S)$ with $F\subseteq\Pi_h$, $S\subseteq\Pi_s$ and $\pi'(F')=\pi'(F)\cup S=F\cup S$. Therefore, we define:

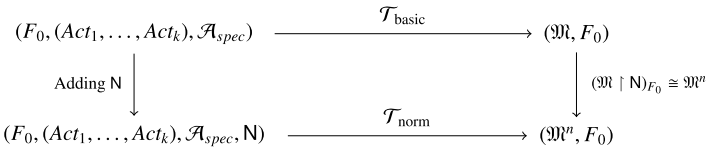

Fig. 13. Diagram showing the relation between a norm update of a generated CGS and a norm generated CGS.

$$G = \begin{cases} F & \text{if } F' = (F, S), \\ F' & \text{otherwise, i.e. } F' \subseteq \Pi_h \end{cases}$$

Furthermore, we note that for all $\vec{\alpha} \in Act_1 \times \ldots \times Act_k$ we have that: $S^* := \mathsf{San}_{\mathsf{N}}(\pi^n(F^n), \vec{\alpha}) = \mathsf{San}_{\mathsf{N}}(\pi'(F'), \vec{\alpha})$ and that in F^n and F' the same action profiles are enabled. We show the claim by considering the three cases how a transition can occur according to Definition 11.

Case $S^* = \emptyset$**.** In that case we have that $o'(F', \vec{\alpha}) = o(G, \vec{\alpha})$, where $o'(F', \vec{\alpha}) \subseteq \Pi_h$ which is generated by $G \xrightarrow{\vec{\alpha}} o'(F', \vec{\alpha}) \in \tau_b$. As the precondition of $\vec{\alpha}$ is independent of soft facts and the first three transition rules of Definitions 16 and 21 are identical for $S^* = \emptyset$, we obtain that $o^n(F^n, \vec{\alpha}) = o'(F', \vec{\alpha})$. The claim follows.

Case $S^* = \{\bot\}$**.** In this case $F^n \xrightarrow{\vec{\alpha}} F^n \in \tau_n$ and $o^n(F^n, \vec{\alpha}) = F^n$ as well as $o'(F', \vec{\alpha}) = F'$. The claim follows because $\pi'(F') = \pi^n(F^n)$.

Case $\{\bot\} \neq S^* \neq \emptyset$**.** We further divide this case according to the first three transition rules of Definitions 16 and 21.

- First suppose that $(Pre, \vec{\alpha}, Post) \in \mathcal{A}_{spec}$ and $F^n \vdash Pre$ (and thus also $G \vdash Pre$). We have that $o^n(F^n, \vec{\alpha}) = (F^n \oplus Post) \otimes S^*$. Analogously, we have that $o'(F', \vec{\alpha}) = (G \oplus Post, S^*)$. We obtain that: $\pi^n(o^n(F^n, \vec{\alpha})) = \pi^n((F^n \oplus Post) \otimes S^*) = \pi^n(((F^n \oplus Post) \backslash \Pi_s) \cup S^*) = \pi^n(((F^n \oplus Post) \backslash \Pi_s)) \cup S^* = \pi^n((G \oplus Post)) \backslash \Pi_s) \cup S^* = \pi^n(G \oplus Post) \cup S^* = \pi'((G \oplus Post, S^*)) = \pi'(o'(F', \vec{\alpha}))$.
- The second case, i.e. $F^n \nvdash Pre$, follows analogously.
- Also the final case, i.e. $(Pre, \vec{\alpha}, Post) \notin \mathcal{A}_{spec}$ for any $Pre, Post \subseteq \Pi_h$, follows analogously. □

As the final result of this section, we show that the concurrent game structures generated by an environment program with and without norms are closely related. In particular, we show that the concurrent game structure generated by a norm-based environment program is isomorphic to the reachable part of the concurrent game structure, which is generated by the corresponding environment program without norm, and updated with the same set of norms. We formally introduce the notions of reachable states and isomorphism before showing the correspondence result.

Definition 23 *(Reachable states, reachable CGS).* Let $\mathfrak{M} = (\mathbb{A}\text{gt}, Q, \Pi, \pi, Act, d, o)$ be a CGS and $q_0 \in Q$. A state $q \in Q$ is said to be reachable from q_0 if there is a path starting in q_0 which also contains q. The set of all states reachable from q_0 in $\mathfrak{M}$ is denoted as $\text{Reachable}(\mathfrak{M}, q_0)$. The reachable part of $\mathfrak{M}$ from q_0, denoted as $\mathfrak{M}_{q_0}$, is the CGS $(\mathbb{A}\text{gt}, Q', \Pi, \pi', Act, d', o')$ where $Q' = \text{Reachable}(\mathfrak{M}, q_0)$, $\pi'(q) = \pi(q)$, $d'_i(q) = d_i(q)$, and $o'(q, \vec{\alpha}) = o(q, \vec{\alpha})$ for all $\vec{\alpha} \in d'_{\mathbb{A}\text{gt}}(q)$, $q \in Q'$ and $i \in \mathbb{A}\text{gt}$.

A model updated by sanctioning norms can yield states of type (q, S). States in norm generated CGSs, on the other hand, have no internal structure; they are plain sets of propositional symbols. We are less interested in such purely syntactic differences and need a way to compare models from a semantic perspective. Therefore, we say that two models $\mathfrak{M}_1$ and $\mathfrak{M}_2$ are *isomorphic* if they are identical besides the names of the states. The next definition captures this formally.

Definition 24 *(Isomorphic models).* Let $\mathfrak{M}_i = (\mathbb{A}\text{gt}_i, Q_i, \Pi_i, \pi_i, Act_i, d_i, o_i)$ for $i \in \{1, 2\}$ be two CGSs. $\mathfrak{M}_1$ and $\mathfrak{M}_2$ are *isomorphic*, written as $\mathfrak{M}_1 \cong \mathfrak{M}_2$, if the following conditions hold: $\mathbb{A}\text{gt}_1 = \mathbb{A}\text{gt}_2$, $\Pi_1 = \Pi_2$, $Act_1 = Act_2$ and there is a bijection $f : Q_1 \to Q_2$ such that:

1. $\pi_1(q) = \pi_2(f(q))$ for all $q \in Q_1$.
2. $d_1(q) = d_2(f(q))$ for all $q \in Q_1$.
3. $f(o_1(q, \vec{\alpha})) = o_2(f(q), \vec{\alpha})$ for all $q \in Q_1$ and $\vec{\alpha} \in d_1(q)$.

Note that we focus on the part of the generated CGS that is reachable from the initial state since the application of regimenting norms in the operational semantics of norm-based environment programs blocks transitions and thus causes some states to become unreachable. Thus, having an environment program $(F_0, (Act_1, \ldots, Act_k), \mathcal{A}_{spec})$ and a set of norms N, we can now show that the concurrent game structure $\mathfrak{M}^n$ generated by $(F_0, (Act_1, \ldots, Act_k), \mathcal{A}_{spec}, \mathsf{N})$ is isomorphic to $(\mathfrak{M} \upharpoonright \mathsf{N})_{F_0}$, which is the part of the concurrent game structure generated by $(F_0, (Act_1, \ldots, Act_k), \mathcal{A}_{spec})$ (i.e., environment program without norms) and updated with N, and is reachable from F_0. This relation is illustrated in the diagram shown in Fig. 13 and formulated in the following theorem.

Theorem 15. *Let* $(F_0, (Act_1, \ldots, Act_k), \mathcal{A}_{spec})$ *be an environment program that* $\mathcal{T}_{basic}$*-generates the pointed concurrent game structure* $(\mathfrak{M}, F_0)$ *with* $\mathfrak{M} = (\mathbb{A}\text{gt}, Q, \Pi, \pi, Act, d, o)$*, and* $(F_0, (Act_1, \ldots, Act_k), \mathcal{A}_{spec}, \mathsf{N})$ *be a norm-based environment program that* $\mathcal{T}_{norm}$*-generates the pointed concurrent game structure* $(\mathfrak{M}^n, F_0)$ *with* $\mathfrak{M}^n = (\mathbb{A}\text{gt}, Q^n, \Pi, \pi^n, Act, d^n, o^n)$*. We define* $\mathfrak{M}' = \mathfrak{M} \upharpoonright \mathsf{N}$ *with* $\mathfrak{M}' = (\mathbb{A}\text{gt}, Q', \Pi, \pi', Act, d', o')$*. Then, we have that* $\mathfrak{M}^n \cong (\mathfrak{M} \upharpoonright \mathsf{N})_{F_0}$*.*

Proof. We define a (partial) function $f : Q^n \to Q'$ by $f(F^n) = F'$ if and only if $\pi^n(F^n) = \pi'(F')$. That f is well defined follows from the fact that for all $F_1, F_2 \in Q'$ we have that $\pi'(F_1) \neq \pi'(F_2)$ whenever $F_1 \neq F_2$. This is because a norm update is only performed once, causing states to be of the form $F \subseteq \Pi_h$ or (F, S) with $F \subseteq \Pi_h$ and $\emptyset \neq S \subseteq \Pi_s$. We show that f constitutes an isomorphism according to Definition 24 between $\mathfrak{M}^n$ and $(\mathfrak{M} \upharpoonright \mathsf{N})_{F_0}$. Therefore, we have to show that f is a bijection (and total) and that the three conditions of Definition 24 are satisfied. Condition 1 is true by the definition of f. Condition 2 is true because both models are uniform by Proposition 9 and Definition 11, and by Proposition 11, respectively. Thus, only the last of these three conditions, i.e. that $f(o^n(q, \vec{\alpha})) = o'(f(q), \vec{\alpha})$ for all $q \in Q^n$ and $\vec{\alpha} \in d^n(q)$, remains to be shown. By definition, both sets of states Q' and Q^n consist of all states which are reachable by some sequence of actions from F_0. Therefore, we show that the claim holds by induction on the length of an action sequence.

Base case. We have that $F_0 \in Q^n \cap Q'$, thus, $f(F_0) = F_0$ is well defined. Moreover, F_0 is reached by the empty action sequence. By Lemma 14 and as $f(F_0) = F_0$ we obtain that $\pi'(o'(F_0, \vec{\alpha})) = \pi^n(o^n(F_0, \vec{\alpha}))$ for any $\vec{\alpha} \in Act_1 \times \ldots \times Act_k$. This implies that $f(o^n(F_0, \vec{\alpha})) = o'(F_0, \vec{\alpha})$ and thus also $f(o^n(F_0, \vec{\alpha})) = o'(f(F_0), \vec{\alpha})$ as desired.

Induction step. We assume that the claim holds for all action sequences of length i. Thus, let $F^n \in Q^n$ and $F' \in Q'$ be two states reached after the same action sequence of length i. By induction, $f(F^n) = F'$ and Condition 3 is satisfied for these states. Let $\vec{\alpha} \in Act_1 \times \ldots \times Act_k$ be an arbitrary action profile. Again, by Lemma 14 we obtain that $\pi'(o'(F', \vec{\alpha})) = \pi^n(o^n(F^n, \vec{\alpha}))$. Consequently, $f(o^n(F^n, \vec{\alpha})) = o'(F', \vec{\alpha})$ is well defined. To establish Condition 3 we consider another arbitrary action profile $\vec{\beta} \in Act_1 \times \ldots \times Act_k$. Because both states are in relation by f we can once more apply Lemma 14 and obtain $\pi^n(o^n(o^n(F^n, \vec{\alpha}), \vec{\beta})) = \pi'(o'(o'(F', \vec{\alpha}), \vec{\beta}))$ which implies that $f(o^n(o^n(F^n, \vec{\alpha}), \vec{\beta})) = o'(o'(F', \vec{\alpha}), \vec{\beta}) = o'(f(o^n(F^n, \vec{\alpha})), \vec{\beta})$. Which shows that Condition 3 is satisfied.

In each inductive step we also showed that f is well defined and total. This implies that f is indeed a bijection as we considered arbitrary action sequences and thus made sure that each state from Q' is reached. □

This theorem shows that the operational semantics of the proposed executable specification languages for environments with and without norms are aligned with the semantics of norms and norm update, as presented in the first part of the paper. This result allows us to apply the proposed abstract mechanism design methodology to analyse the enforcement effect of norms in executable environment programs.

Concluding remarks

In this section, an executable specification language for the implementation of multi-agent environments was presented. The language includes constructs to specify the initial state of multi-agent environments as well as the specification of action profiles in terms of pre- and postconditions. The operational semantics of the proposed specification language was presented and its relation with concurrent game structures was established. An execution of an environment specification initializes a multi-agent environment, which is subsequently modified by the performance of the agents' actions and according to the action specifications. Subsequently, the specification language was extended with norms and sanctions, its operational semantics was presented, and its relation with concurrent game structures that are updated with norms was established. The operational semantics ensures that the norms are enforced by means of regimentation or sanctions. An execution of a norm-based environment specification initializes a multi-agent environment and effectuates agents' actions in the environment based on the action specifications, the norms, and their corresponding sanctions. The agents are assumed to be aware of the norms and their enforcement. They are also assumed to autonomously decide whether to comply with the norms, or to violate them and accept the consequences. The presented executable specification language with norms can be used in various application domains such as traffic or business process management, where the behaviour of autonomous agents should be externally controlled and coordinated to be aligned with some laws, rules, policies, or requirements. The next section reports on an application of the extended specification language in traffic simulation, where traffic laws (norms) are enforced to reduce traffic congestion in a ramp merging scenario. We argue that although the executable specification language is useful for the implementation of traffic simulations, there are specific issues that should be resolved before we can apply the proposed norm-based mechanism design methodology to such applications.

6. Applying norm-based specification language in traffic simulation

A characteristic feature of the proposed specification language for norm-based multi-agent environments is the modularity of norms in the sense that norms can be programmed as a separate module isolated from the specification of actions and (initial) states of multi-agent environments. This feature allows us to implement different sets of norms and to compare their enforcement effects in one and the same multi-agent environment. The use of the proposed specification language is already illustrated by the running example that is presented in previous sections (see Example 17). In this section we present a more complex and realistic application of the proposed specification language.

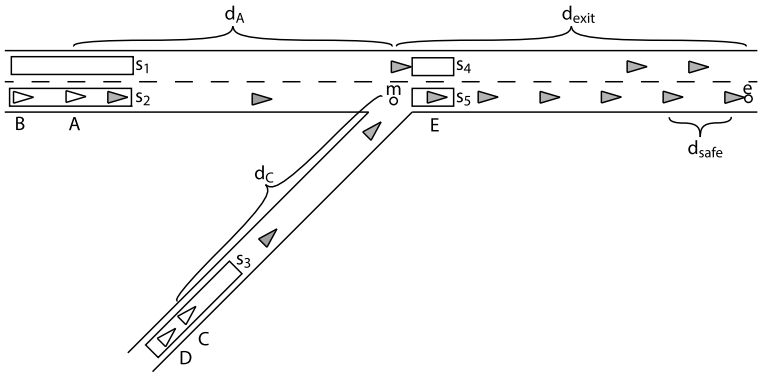

Fig. 14. Ramp merging traffic scenario.

This application concerns the development of norm-based traffic environments for SUMO (Simulation of Urban MObility) [12]. SUMO [46] is a traffic simulation platform that supports the modelling of traffic, including cars, public transport and pedestrians. In this application, SUMO is extended to simulate traffic scenarios, where norms and traffic laws are explicitly specified as input and enforced during the simulation runs. In particular, SUMO is extended with a norm-based traffic controller module that monitors the simulated traffic by continuously extracting relevant traffic information such as the position and speed of the simulated cars from the SUMO platform, instantiates the given input set of traffic norms to generate traffic directives for the observed cars, communicates traffic directives to the cars, and imposes payment sanctions on the cars that violate their traffic directives. In addition to the traffic controller module, the standard SUMO car model that is responsible for the actual behaviour of individual cars, is extended to allow individual cars to incorporate norms in their driving behaviour.

The SUMO extension [12] is used to simulate a ramp merging traffic scenario. A schematic representation of the ramp merging scenario is illustrated in Fig. 14. In this figure, triangles represent cars that drive from left to right and rectangles s_1 to s_5 are sensors that observe the position and speed of cars at various points of the roads. There are two important points on the road: m is the point where the roads merge and e is the ending point of the traffic scenario. In order to manage traffic at the merging point m, observed cars at sensor s_1, s_2 and s_3 receive traffic directives from the traffic controller. The directive for a car is generated based on the given set of traffic norms and consists of a velocity and a fine that will be imposed if the directive is not followed by the car. In this figure, white cars have not received their directives from the traffic controller, while grey cars have received their directives.

An example of a traffic norm used in this scenario is $(x \wedge y, \mathcal{A}_{(v_x,v_y)}, \{fine_z\})$ to be read as "cars x and y, observed simultaneously by sensor s_2 and s_3, are prohibited to have velocities other than respectively v_x and v_y at the merging point m to avoid a fine of z Euro". For this traffic norm, the set of prohibited velocities $\mathcal{A}_{(v_x,v_y)} = \{(v_1, v_2) \mid v_1 \neq v_x \text{ and } v_2 \neq v_y\}$[13] represents the obligation that cars x and y should have velocities v_x and v_y, respectively, at the merging point m. The traffic controller instantiates the input norm by determining x, y, v_x, v_y, and $fine_z$ based on the observed cars and the properties of the current traffic state such as traffic density on the roads. The velocities of the observed cars are determined in such a way that cars arrives at m with a safe distance d_{safe} from each other given the current traffic density. The fine of z Euro will be imposed on car x (respectively y) if its velocity v_x (respectively v_y) at m is not realised. Based on the instantiated norm, the traffic controller sends to each observed cars a corresponding directive. For example, the traffic controller sends to the observed car x the directive $(v_x, fine_z)$, which should be read as "car x should have velocity v_x at the merge point m to avoid the fine of z Euro". For this simulation scenario, the traffic controller generates also directives for the cars that are observed by one of the two sensors s_2 or s_3, even if there is no car simultaneously observed at the other sensor.

In this traffic simulation, a car decides which action (e.g., which velocity on which lane) to perform and whether to follow or ignore the received directives based on its information and preferences. In particular, the car model is designed by means of an action selection function that selects an action that maximizes the car's utility. The action selection function is defined in terms of the car's internal state (including its current velocity, position, and the received directive), an expected arrival time function that, given the current traffic situation, determines the impact of a velocity action on the arrival time at the car's destination (for simplicity it is assumed that all cars have the same destination), an action effect function (that determines the expected consequence of a velocity action on its internal state), a sanction grading function (mapping fines to real values reflecting the utility of a fine), and an arrival grading function (mapping an arrival time to real values reflecting the utility of the given arrival time at the car's destination). The internal state of the car together with the expected arrival time function and the action effect function constitute an agent's information about themselves and the traffic situation

[13] We assume that v_1 and v_2 are taken from some sensible finite set of velocity values.

including other cars. The sanction grading function and the arrival grading function constitute the preference of the cars. The further details of the car model can be found in [12].

The objective of this simulation was to investigate the impact of the enforcement of various norm sets on traffic situation in the ramp merging scenario for a different population of cars. Two types of cars were distinguished: leisure and business cars. These two types were implemented by the sanction and arrival time grading functions. The sanction grading function for a leisure car evaluates a sanction as having a higher impact on the car's utility compared to the same function for a business car. Conversely, the arrival time grading function for a leisure car evaluates a late arrival time as having less impact on the car's utility compared to the same function for a business car. The simulation results show that the number of norm violations, and therefore traffic congestion, decreases as the severity of fines increases. In particular, they show that norm violations do not occur when proper fines (i.e., fines that match the cars' preferences) are imposed.

The formal connection between the proposed abstract mechanism design methodology and the specification language, as established in Proposition 14, may suggest that the results of the mechanism design analysis of norms can be related to the results of the implementation of norms in multi-agent simulations. Such a relation can be used to justify the simulation results by means of mechanism design explanations, or vice versa, to verify the results of the mechanism design analysis by means of simulations. For example, if a mechanism design analysis of the ramp merging scenario shows that a set of traffic norms implements the system designer's objective to avoid traffic congestion (i.e., to avoid simultaneous arrival of cars at the merging point m) assuming that cars follow their Nash equilibrium strategies, then one can use the specification language to simulate the scenario in order to verify whether the enforcement of the traffic norms avoids traffic congestion.

However, connecting theoretical results obtained by a mechanism design analyses and by experimental results obtained by running agent-based simulations is not straightforward and requires further considerations. For example, one could argue that the reported experimental results from the traffic simulation in SUMO can be used to claim that an observed reduction of traffic congestion is due to the optimality of norms in the sense that the norms implement the objective of avoiding traffic congestion in Nash equilibrium. However, such claims could only be justified if the simulated cars were capable of strategic reasoning, which is not the case in the reported traffic simulation experiment. The ability of strategic reasoning for cars is not supported by our extension of the SUMO platform and requires further extensions. This is due to the fact that the preferences of the cars are not accessible to each other such that the cars do not share the structure of the game and are thus unable to reason strategically. This implies that the simulations setting in SUMO is not yet rich enough to establish a connection to our mechanism design setting.

It should also be noted that our formal mechanism design methodology can now be applied to analyse only one snapshot of the traffic simulation, i.e., to analyse the behaviour of cars that arrive simultaneously at the sensor positions. Such a snapshot of the ramp merging scenario constitutes a game setting that is quite comparable with the setting of our running example (the narrow part of the road in the running example is the merging point of the ramp merging scenario). In order to connect the mechanism design methodology and the experimental results of the simulations, which consists of a continuous stream of cars, one could model the simulation as a sequence of games. Although this may be a reasonable suggestion, one needs to investigate how to model and analyse the change and development of the traffic state in consecutive game settings. For example, a high stream of cars may necessarily cause the creation of traffic congestion at the merging point, which changes the state of the traffic and therefore the structure of the consecutive games. We believe that a profound connection between mechanism design and simulation settings is a challenging future research direction.

7. Related work

There have been many proposals on abstract and executable models for norms and norm enforcement in multi-agent systems. However, unlike our work, these proposals focus either on abstract models of norm enforcement or on executable models, ignoring the connection between them. Despite this key difference, our abstract and executable model differ from existing abstract and executable models, respectively. In the following, we first compare our abstract model for norm and norm enforcement to existing abstract models, and then compare our executable specification language with existing executable models for norms and norm enforcement.

Our abstract model of norms and norm enforcement is closely related to [1] and [65], though they consider a norm from a semantic perspective as a set of transitions instead of a syntactic expression as in our case. In particular, they use labelled Kripke structures as multi-agent models, supposing that each agent controls a specific number of transitions. The enforcement of a norm is then considered as the deactivation of specific transitions. In our proposal, we distinguish between regimenting and sanctioning norms, which is also one of the main conceptual differences besides the mechanism design perspective we follow. The enforcement of the regimenting norms is similar to the deactivation of transitions as in [1,65], but the enforcement of sanctioning norms may create new states resulting from a relabelling by soft facts and thereby new transitions. In this sense, our approach is different as the enforcement of norms may change the underlying multi-agent model with new states and transitions. From a conceptual point of view the agent can still perform the action, the physical structure encoded by means of hard facts remains intact, but may have to bear the imposed sanctions depending on the agents' preferences. Note also, the change in the underlying transition system may only affect some agents, namely those which use relevant soft facts in their preferences. Another difference to our work is that the outcome of norm enforcement is assumed to be independent of the preferences where we consider a more general setting captured in terms of normative behaviour functions. 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 implement and verify norm-based multi-agent environment by means of executable specifications. The work presented in [3] also assumes that the designer has multiple objectives the value of which is determined by the (de)activated transitions in the transition system. The authors show how to compactly represent the designer's objectives by (a set of) CTL formulae each assigned a feature value. Then, the utility of a social law is the sum of all feature values of the satisfied CTL formulae minus the costs to implement the social law. The authors give an algorithm to compute optimal social laws as well as complexity results. The focus of the work is on computing a good social law from the designer's perspective, not considering agents' objectives at first place (which is a key concern in our setting).

Fitoussi and Tennenholtz [34] put forward a formal framework to engineer and study social laws in multi-agent systems. A social law restricts the set of agents' actions. The authors investigate two properties of social laws: minimality and simplicity. Minimality refers to the number of restrictions imposed on the system, where simplicity refers to the capabilities of agents. Simpler laws can be followed more easily by simpler agents. The authors consider complexity issues about the existence of appropriate norms and study their properties. The setting is quite abstract, using strategies similar to those known from normal form games, whereas we start from a transition system usually requiring multiple step strategies. Also, the authors consider two specific types of goals, liveness and safety goals, whereas we allow arbitrary LTL-formulae to specify agents' preferences.

Endriss et al. [30] and Wooldridge et al. [69] propose taxation schemes to incentivize agents to change their behaviour such that the emerging behaviour is stable and shows desirable properties in line with the system specification. In particular, the computational complexity of the weak and strong implementation problems is investigated. We have drawn inspiration from these problems in the present article. A difference with our work is that they study these problems in the context of Boolean games, where we consider a strategic setting in which agents act in a temporal setting. Instead of taxation schemes we follow a norm-based approach, and use techniques of mechanisms design to specify the system designer's objectives and analyse their implementability. We believe that this approach is quite flexible and allows to model more realistic settings in which the system designer may not know the agents' preferences and should therefore consider a set of possible preferences.

A different avenue of work in this area focuses on norm monitoring in the context of imperfect monitors, e.g., [20,8]. In the present paper, we have assumed that monitors are perfect in the sense that the norm enforcement mechanism can detect and respond to all norm violating behaviours. It should be clear that any work on norm enforcement either implicitly or explicitly assumes that the behaviour of agents is monitored and evaluated with respect to a given set of norms. Our assumption that monitors are perfect is reflected by the fact that updating a multi-agent model (i.e. CGS) with a set of norms covers possible (violating) behaviour. Moreover, although [20] and [8] consider norms syntactically like us, they use LTL-formulae as norm representation to refer to good/bad behaviour. We have considered less expressive norms in order not to complicate the main message of our approach unnecessarily. We believe that our general approach can be instantiated with more expressive norms as well.

Another line of related research concerns the issue of norm synthesis. Shoham and Tennenholtz [59,60], have discussed the problem of off-line design of a set of norms in order to constrain the behaviour of agents and to ensure the overall objectives of the multi-agent system. In this work and similar to our approach, the structure of multi-agent systems is required and the norms are generated at design time. In line with this tradition, Morales et al. [53,52] consider the problem of on-line design of a set of norms to constrain and steer the behaviour of agents. In both off-line and on-line approaches, the overall objectives of multi-agent systems are guaranteed by assuming that agents are norm-aware and comply with the generated norms. In this sense norms are considered as being regimented in multi-agent systems. Moreover, in contrast to our work, these approaches neither provide a game theoretic analysis of norms nor consider the generation of norms with sanctions in the context of agents' preferences. However, we believe that the concepts such as effective, necessary, and concise norms as introduced by [53,52] can also be used in off-line norm synthesis approaches such as ours. Following the tradition of Shoham and Tennenholtz, Boella and van der Torre [15] consider the problem of norm enforcement by distinguishing the choice of off-line designed stable social laws from the choice of control systems. A control system is explained to be responsible for monitoring and sanctioning of norm violations. Although the functionality of their proposed notion of control system is similar to the functionality of our notion of norm-based mechanism, there are some fundamental differences between these approaches. For example, in our approach the behaviour of a multi-agent system is modelled by a concurrent game structure while they consider a multi-agent system in an abstract way as a one-shot game, norms in our approach are explicit and enforced by an update mechanism while they consider norms as integrated in the structure of the games and enforced by a special agent called normative system that selects which game is going to be played, and finally they consider the notion of (quasi-)stable social laws while we consider norms from a mechanism design perspective as implementing a social choice function in Nash equilibria.

Our work differs also from verification approaches to norm-based systems or protocols [33,7,27,5]. Of course, one can consider and exploit our work as an approach to verify the impact of norm enforcement on agents' behaviour in the sense that our approach can be used to check the influence of norm enforcement on agents' behaviour. However, in contrast to our approach, the mentioned work focuses on different types of norms and norm enforcement mechanisms, and does not provide any game theoretic tool to analyse the impact of norms on the behaviour of rational agents. In particular, in [33] norms are defined in terms of communication actions and enforced by means of regimentation, while [7] and [27] consider norms as state-based obligations/prohibitions enforced by means of both regimentation and sanctions. The approach pre-

sented by [5] focuses mainly on the protocol verification and aims at providing a mechanism that can be used, for example by the agents, to decide whether following a protocol guarantees their objectives without any norm violation.

In the literature of multi-agent systems various proposals focus on the issue of the practical implementation of norms and norm enforcement. They consider norms and norm enforcement mechanisms in broader contexts such as institutions, organisations, or coordination environments. Examples are electronic institutions such as ISLANDER/AMELI [33,32], organisational models such as $\mathcal{M}$OISE$^+$/$\mathcal{S}$–$\mathcal{M}$OISE$^+$ [42,43] and OperettA [4], coordination models such as ORG4MAS [41], the norm-based framework proposed by [24], the law-governed interaction proposed by [51], and the norm enforcement mechanism proposed by [35]. The focus on these proposals is primarily on the development of norm-based multi-agent system rather than devising special purpose programming languages to implement norms and norm enforcement. Moreover, the lack of explicit formal syntax and (operational) semantics for norm-related concepts in these approaches makes it difficult, if not impossible, to relate them to the existing abstract models for norms and norm enforcement. These approaches are concerned with agents' interaction, use different norm types, or focus on norm regimentation only. In the following, we discuss further details of some of these approaches.

ISLANDER [31] is one of the early modelling languages for specifying institutions in terms of institutional rules and norms. In order to interpret institution specifications, an execution platform, called AMELI, has been developed [32]. This platform implements an infrastructure that, on the one hand, facilitates agent participation within the institutional environment supporting their communication and, on the other hand, enforces the institutional rules and norms. The distinguishing feature of ISLANDER/AMELI is that it does not allow any agent to violate norms, i.e., norms are regimenting. Moreover, norms in [32], but also in [36] and [61], are action-based and prescribe actions that should or should not be performed by agents.

Another related approach is $\mathcal{M}$OISE$^+$ [43], where a modelling language is proposed to specify multi-agent systems through three organisational dimensions: structural, functional, and deontic. The relevant dimension for our work is the deontic dimension that concerns concepts such as obligations and prohibitions. Different computational frameworks have been proposed to implement and execute $\mathcal{M}$OISE$^+$ specifications, e.g., $\mathcal{S}$–$\mathcal{M}$OISE$^+$ [42] and its artifact-based version ORG4MAS [41]. These frameworks are concerned with norms prescribing states that should be achieved/avoided. $\mathcal{S}$–$\mathcal{M}$OISE$^+$ is an organisational middleware that provides agents access to the communication layer and the current state of the organisation. This middleware allows agents to change the organisation and its specification, as long as such changes do not violate organisational constraints. In this sense, norms in $\mathcal{S}$–$\mathcal{M}$OISE$^+$ can be considered as regimenting norms. ORG4MAS uses organisational artifacts to implement specific components of an organisation such as group and goal schema. In this framework, a special artifact, called reputation artifact, is introduced to manage the enforcement of norms.

There are two proposals that specifically aim at implementing norms and norm enforcement. The first proposal, presented in [64] and [27], is a norm-based executable specification language designed to facilitate the implementation of software entities that exogenously control and coordinate the behaviour of agents by means of norms. Similar to our approach, norms in this proposal can be either sanctioning or regimenting norms. Also, similar to our approach, they come with an operational semantics such that executable specifications can be formally analysed by means of verification techniques [11]. However, in contrast to the approach presented in the present paper, they consider state-based norms such as obligation or prohibition of states and ignore action-based norms. This proposal comes with an interpreter, called 2OPL, which initiates a process that continuously monitors agents' actions (i.e., communication and environment actions), determines the effect of the observed actions based on the action specifications, and enforces norms when necessary. We plan to extend 2OPL such that it can interpret and execute action-based norms as presented in the present paper. The second proposal, called JaCaMo [16], aims at supporting the implementation of organisational artifacts, which are responsible for the management and enactment of the organisation. An organisational artifact is implemented by a program which implements a $\mathcal{M}$OISE$^+$ specification. A translation of $\mathcal{M}$OISE$^+$ specifications into norm-based programs is described by [44]. In contrast to our approach, the sanctions in JaCaMo are actions that are delegated to some agents and there is no guarantee that the agents will eventually perform the actions. In particular, the violation of norms is detected by organisational artifacts after which organisational agents have to deal with those violations.

We conclude this section by the following observation. We have assumed that agents are norm aware in the sense that agents follow their preferences and choose optimal behaviours in order to maximize their utilities. The norm awareness is reflected in our approach by 1) defining agents' preferences in terms of specific behaviour and whether the agents incur sanctions, and 2) by applying the equilibrium analysis to characterize the behaviour of rational agents under norm enforcement. However, we did not focus on how individual agents reason to choose their optimal behaviours as studied, for example, by [66] and [6]. In contrast to [66] and [6], we abstract over the specific reasoning schemes of individual agents and assume that whatever reasoning schemes individual agents use, they will always act rationally according to game theoretic concepts, more precisely according to Nash equilibria.

8. Conclusions, discussion, and future work

Our work focuses on norms and norm enforcement in multi-agent systems. We proposed a formal methodology that allows analysing norms and norm enforcement from a mechanism design perspective, and a programming model for implementing them. Using game theoretic tools we showed that the enforcement of norms can change the behaviour of rational agents using regimentation and sanctions. It is also shown that our presented programming model is aligned with the abstract model such that our developed game theoretical tools can be applied to analyse norm-based environment programs.

Specifically, we proposed norm-based mechanism design as a formal methodology for analysing norm-based environment programs. We showed how to abstract from a particular environment specification language and how to apply methods from mechanism design to verify whether the enforcement of a set of norms on a multi-agent environment agrees with the behaviour of rational agents that the system designer expects. More precisely, we introduced normative behaviour functions for representing the "ideal" behaviour of multi-agent environments with respect to different sets of agents' preferences. The latter enabled us to apply concepts from game theory to identify agents' rational behaviour. These formal ideas can now be used to verify whether the enforcement of a set of norms is sufficient to motivate rational agents to act in such a way that their behaviour become aligned with that described by the normative behaviour function.

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

Another direction for 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 to specify its preference. 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 normative system such that $\Pi_{NF} \cap \Pi_a = \emptyset$. Another alternative is to take on a more game theoretic point of view in the line with [1] and [23]. 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 investigated the problem, given a CGS $\mathfrak{M}$ and a set of agents' preferences $\mathcal{P}refs$, and a normative behaviour function f, whether there is a normative system N which $\mathcal{NE}$-implements f over $\mathfrak{M}$, q and $\mathcal{P}refs$. In future work it would be interesting to identify settings in which such normative systems can be constructed efficiently. We also plan to extend our analysis to other implementability notions apart from Nash equilibria in more detail, e.g. dominant strategy equilibrium implementability.

Finally, yet another interesting direction for future research is to investigate core properties of classical mechanism design in our norm-based setting, including budget balanced and individual rational mechanisms. We note that already the interpretation of these properties in our setting is interesting. For example, in the case of individual rationality it is not clear what it means for an agent "not to take part" in the mechanism/in the multi-agent systems. This may require a shift to an open MAS where agents can leave an join the system.

Acknowledgement

We thank the anonymous reviewers for their extensive and valuable comments which significantly improved the paper.

Appendix A. Quantified Boolean satisfiability problem

Our hardness proofs reduce validity of Boolean quantified formulae to the implementation problems. We consider fragments of the Quantified Boolean Satisfiability problem (**QSAT**), a canonical **PSPACE**-complete problem. Restricting the number of quantifier alternations of **QSAT** yields subproblems which are complete for different levels of the polynomial hierarchy. The problem class **QSAT**$_i$ starts with an existential quantifier and allows for $i-1$ quantifier alternations; similarly, $\forall$**QSAT**$_i$-formulae begin with an universal quantifier, $i = 1, 2, \ldots$. The previous problems are $\mathbf{\Sigma_i^P}$ and $\mathbf{\Pi_i^P}$-complete, respectively. In the formal definition we write $Q\,X$ for $Q\,x_1 \ldots Q\,x_n$ for a set $X = \{x_1, \ldots, x_n\}$ of propositional variables and $Q \in \{\forall, \exists\}$.

Definition 25 (**QSAT**$_i$ [55]). The **QSAT**$_i$ problem is defined as follows.
Input: A quantified Boolean formula (**QBF**) $\phi = \exists X_1 \forall X_2 \ldots Q_i X_i\ \varphi$ where φ is a Boolean formula in negation normal form (nnf) (i.e., negations occur only at the propositional level) over disjoint sets of propositional variables $X_1, \ldots, X_i$ and $i \geq 1$ where $Q_i = \forall$ if i is even, and $Q_i = \exists$ if i is odd. ϕ does not contain any free variables.
Output: True if $\exists X_1 \forall X_2 \ldots Q_i X_i\ \varphi$ is valid, and false otherwise.

The problem $\forall$**QSAT**$_i$ is defined analogously but formulae ϕ start with a universal quantifier and then alternate between quantifier types.

By abuse of notation, we refer to a **QBF** ϕ that satisfies the structural properties required by the problem class **QSAT**$_i$ and $\forall$**QSAT**$_i$ simply as **QSAT**$_i$-formula and $\forall$**QSAT**$_i$-formula, respectively.

A *truth assignment* or *valuation* for a set of variables X is a mapping $v : X \to \{t, f\}$. Given a Boolean formula φ over variables X and a truth assignment v over $Y \subseteq X$ we denote by $\varphi[v]$ the formula obtained from φ where each $y \in Y$ is replaced by $\bot$ (falsum) and $\top$ (verum) if $v(y) = f$ and $v(y) = t$, respectively. For two truth assignments v_1 and v_2 over X and Y, respectively, with $X \cap Y = \emptyset$ we use the notation $v_1 \circ v_2$ to refer to the induced truth assignment over $X \cup Y$; analogously for more than two truth assignments. Using this notation a formula $\exists X_1 \forall X_2 \ldots Q_i X_i\ \varphi$ is true iff there is a

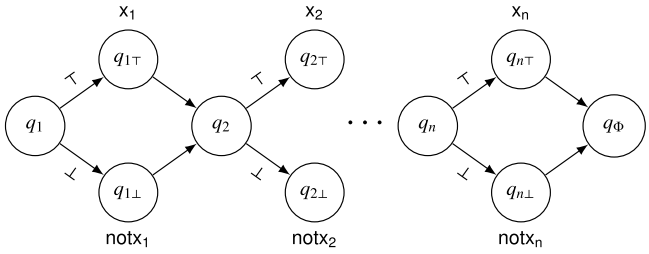

Fig. B.15. Construction of the concurrent game structure for QSAT: value choice section.

truth assignment v_1 over X_1 such that for all truth assignments v_2 over X_2 etc. the Boolean formula $\varphi[v_1 \circ \ldots \circ v_i]$ is valid. For further details, we refer to [55].

Appendix B. Proofs to implementation problems: verification

B.1. Hardness of weak implementation problem: Proposition 4

We show that the membership problem "$\mathsf{N} \in \mathsf{WI}_{\mathcal{NE}}(\mathfrak{I}, q, f)$" is $\mathbf{\Sigma_2^P}$-hard by reducing $\mathbf{QSAT}_2$ to it. In the following we consider an instance of $\mathbf{QSAT}_2$ of the form

$$\phi \equiv \exists\{x_1, \ldots, x_m\}\forall\{x_{m+1}, \ldots, x_n\}\varphi(x_1, \ldots, x_n)$$

where $X_1 = \{x_1, \ldots, x_m\}$ and $X_2 = \{x_{m+1}, \ldots, x_n\}$. The reduction consists of three main steps:

1. We encode ϕ as a two-player CGS $\mathfrak{M}^\phi$ and show that ϕ is satisfiable if, and only if, player one has a winning strategy in $\mathfrak{M}^\phi$ to achieve a given property (Lemma 16).
   We use results from [22] where it was shown that the satisfiability of a **QBF** ϕ can be reduced to model checking a two-player CGS such that one of the players, the *verifier* **v**, has a winning strategy to enforce a (fixed) formula which is constructed from ϕ, against all strategies of the other player, the *refuter* **r**, if and only if, ϕ holds.
2. We construct a preference profile $\vec{\gamma} = (\gamma_{\mathbf{v}}, \gamma_{\mathbf{r}})$ such that a winning strategy of the verifier in $\mathfrak{M}^\phi$ is part of a Nash equilibria in $\Gamma(\mathfrak{M}^\phi, q, (\gamma_{\mathbf{v}}, \gamma_{\mathbf{r}}))$ if, and only if, ϕ is satisfiable.
3. We show that the existence of such a specific Nash equilibria can be answered by testing membership in the weak implementation problem.

In the following we assume that the $\mathbf{QSAT}_2$-formula ϕ given above is fixed, including the indexes m and n and that it is in negated normal form. Moreover, in the following it is important that X_1 and X_2 are non-empty. This can be assumed without loss of generality.

B.1.1. The model $\mathfrak{M}^\phi$

We describe the construction the CGS $\mathfrak{M}^\phi$ from formula ϕ. The idea of the construction in [22] is that the *verifier* **v** (controlling existential variables and disjunctions) and the *refuter* **r** (controlling existential variables and conjunctions) firstly choose truth values of the variables they control. This is illustrated in Fig. B.15.

For example, if **v** plays $\top$ in q_2 a state labelled x_2 is reached; otherwise, a state labelled notx_2. This represents that variable x_2 is assigned true or false, respectively. This part of the model is called *value choice section* and consists of states

$$Q_1 = \{q_i \mid i = 1, \ldots, n\} \cup \{q_{iv} \mid i = 1, \ldots, n \text{ and } v \in \{\bot, \top\}\}.$$

States q_i with $1 \leq i \leq m$ are controlled by **v**, states with $m+1 \leq i \leq n$ are controlled by **r**. Afterwards, both agents simulate the game theoretic semantics of propositional logic. Player **v** tries to make the formula true (thus controls disjunctions) and **r** tries to falsify the formula (thus controls conjunctions). This part of the model corresponds to the parse tree of a formula, see Fig. B.16. For the formal definition we need additional notation. First, we use $\mathsf{sf}(\phi)$ to denote the set of subformulae of ϕ. For every formula $\psi = \psi_1 \circ \psi_2$ with $\circ \in \{\wedge, \vee\}$ we use $L(\psi) = \psi_1$ and $R(\psi) = \psi_2$ to refer to the left and right subformula of ϕ, respectively. If $\psi = \neg\psi'$ and ψ is not a literal[14] we define $L(\psi) = R(\psi) = \psi'$. This allows to use a sequence of L's and R's, i.e. an element from $\{L, R\}^*$, to uniquely refer to a subformula where ϵ refers to ϕ itself. We refer to such a sequence as *index*. For a formula ϕ we use $\mathsf{ind}(\phi) \subseteq \{L, R\}^*$ to denote the set of all possible indexes wrt. ϕ. By slight abuse of notation, we denote the subformula referred to by such an index also by $\mathsf{sf}(i)$ where $i \in \{L, R\}^*$. Note, that different indexes can refer to the same subformula. For example, given the formula $(x_1 \wedge x_2) \vee (x_2 \wedge \neg x_1)$ we have that $\mathsf{sf}(LR) = \mathsf{sf}(RL) = x_2$. Note, as we stop at the literal level, the indexes RRL and RRR are not contained in $\mathsf{ind}((x_1 \wedge x_2) \vee (x_2 \wedge \neg x_1))$.

[14] We stop at the literal level as it simplifies our construction.

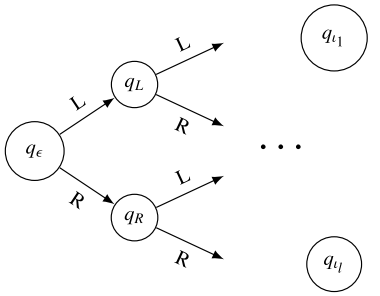

Fig. B.16. CGS for QSAT: formula structure section where $\mathsf{sf}(\iota_1), \ldots, \mathsf{sf}(\iota_l)$ are all literals.

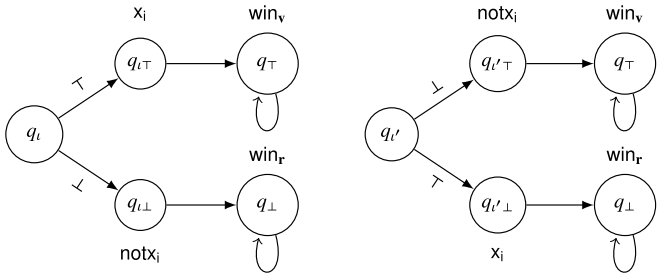

Fig. B.17. CGS for QSAT: sections of literals where $\mathsf{sf}(\iota) = x_i$ and $\mathsf{sf}(\iota') = \neg x_i$.

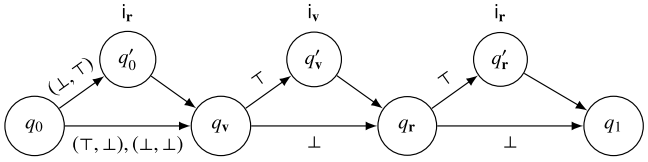

Fig. B.18. Small gadget which can be used by the two players to make their strategy inconsistent.

Given this notation, the *formula structure section* of the model consists of states

$$Q_2 = \{q_\iota \mid \iota \in \mathsf{ind}(\phi),\ \mathsf{sf}(\iota) \text{ is not a literal}\}.$$

For every index ι where $\mathsf{sf}(\iota) = \psi = \psi_1 \circ \psi_2$ with $\circ \in \{\wedge, \vee\}$ one of the players chooses $L(\psi) = \psi_1$ or $R(\psi) = \psi_2$. If $\circ = \vee$ the verifier $\mathbf{v}$ chooses the subformulae; otherwise the refuter does. The "semantic game" between both players ends up in a literal, modelled by a literal state. The *section of literals* is built over states

$$Q_3 = \{q_\iota \mid \iota \in \mathsf{ind}(\phi),\ \mathsf{sf}(\iota) \text{ is a literal}\}$$

for each index ι corresponding to a literal l in Φ, we have that the state q_ι is controlled by the owner of the Boolean variable x_i in l (i.e. $l = x_i$ or $l = \neg x_i$). As in the value choice section, the owner of that state chooses a value ($\top$ or $\bot$) for the underlying variable (not for the literal!) which leads to a new state of the *evaluation section*. These states are denoted by

$$Q_4 = \{q_{\iota v} \mid \iota \in \mathsf{ind}(\phi),\ \mathsf{sf}(\iota) \text{ is a literal and } v \in \{\top, \bot\}\} \text{ and } Q_5 = \{q_\top, q_\bot\}.$$

A state $q_{\iota v}$ with $\mathsf{sf}(\iota) = x_i$ is labelled with the proposition $\mathsf{x_i}$ if $v = \top$ and with $\mathsf{notx_i}$ if $v = \bot$; similarly, $q_{\iota v}$ with $\mathsf{sf}(\iota) = \neg x_i$ is labelled with the proposition $\mathsf{x_i}$ if $v = \bot$ and with $\mathsf{notx_i}$ if $v = \top$. That is, the label $v \in \{\top, \bot\}$ models the evaluation of the literal and not of the underlying variable. These states shall be used to ensure that a strategy induces a truth assignment, as further explained below. Then, the system proceeds to the winning state $q_\top$ (labelled with the proposition $\mathsf{win_v}$) if the valuation of x_i makes the literal $\mathsf{sf}(\iota)$ true, and to the losing state $q_\bot$ (labelled with the proposition $\mathsf{win_r}$) otherwise – see Fig. B.17 for details. Finally, we need two special gadgets to ensure that the reduction works. First, we connect a state q_d to the initial state q_0. This state will be used to ensure the existence of some Nash equilibrium. Secondly, we need to give the players a possibility to mark specific strategies as *inconsistent*, for reasons which will become clear below. Therefore, we insert a small substructure as shown in Fig. B.18 in-between the start state q_0 and q_1, the beginning of the value-choice section. The whole construction is illustrated in Fig. B.19. We refer to the CGS just constructed as $\mathfrak{M}^\phi$. The formal definition is given next.

Definition 26 *(Model $\mathfrak{M}^\phi$).* Let **QSAT**$_2$-formula $\phi \equiv \exists\{x_1, \ldots, x_m\}\forall\{x_{m+1}, \ldots, x_n\}\varphi$ be given and in negated normal form. We define $X_\mathbf{v} = \{x_1, \ldots, x_m\}$ and $X_\mathbf{r} = \{x_{m+1}, \ldots, x_n\}$. The CGS $\mathfrak{M}^\phi = (\mathbb{A}\mathrm{gt}, Q, \Pi, Act, \pi, d, o)$ is defined as follows:

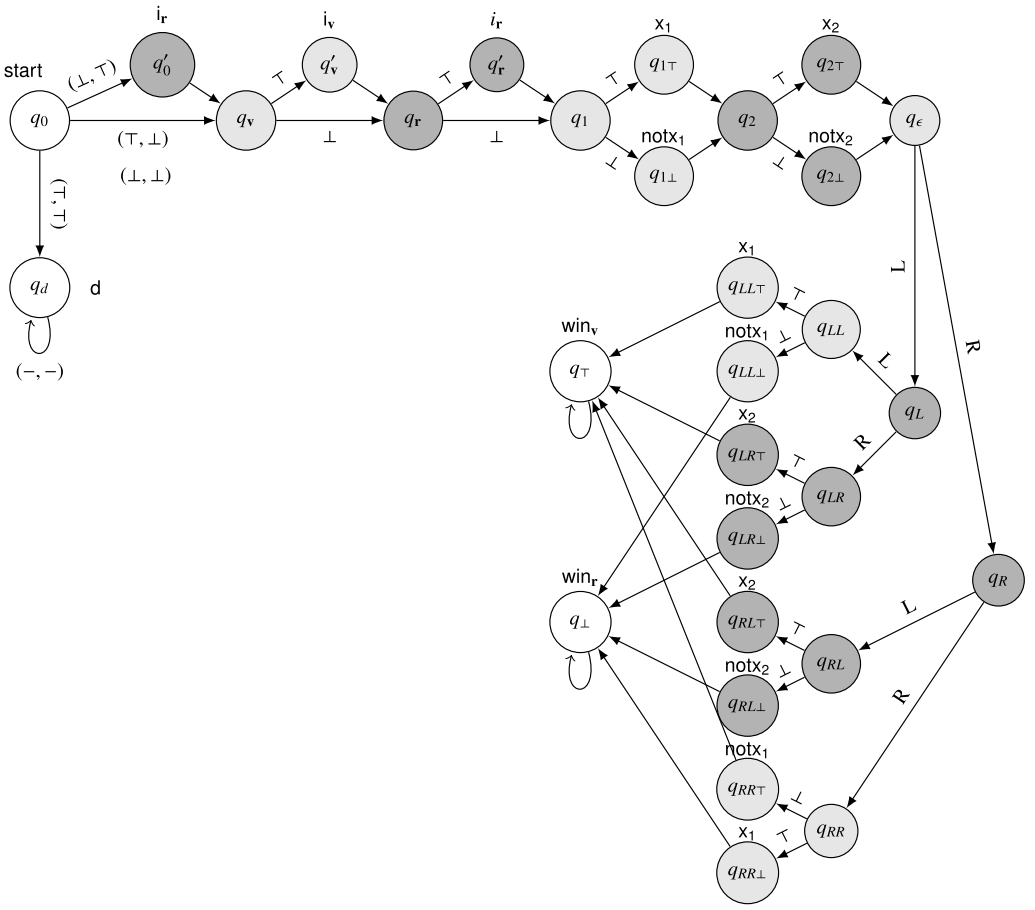

Fig. B.19. Concurrent game structure for the **QSAT**$_2$ instance $\Phi_1 \equiv \exists x_1 \forall x_2 (x_1 \wedge x_2) \vee (x_2 \wedge \neg x_1)$. We have that $\mathsf{ind}(\phi_1) = \{\epsilon, L, R, LL, LR, RL, RR\}$. In particular, we have that: $\mathsf{sf}(\epsilon) = (x_1 \wedge x_2) \vee (x_2 \wedge \neg x_1)$, $\mathsf{sf}(L) = x_1 \wedge x_2$, $\mathsf{sf}(R) = x_2 \wedge \neg x_1$, $\mathsf{sf}(LL) = x_1$, $\mathsf{sf}(RR) = \neg x_1$ etc. "Light gray" states are owned by the verifier; "dark grey" states are owned by the refuter. "White states" belong to no player. A transition outgoing from a state controlled by $\mathbf{v}$ (resp. $\mathbf{r}$) labelled with an action α corresponds to an action profile $(\alpha, -)$ (resp. $(-, \alpha)$). Transitions without label correspond to an action profile $(-, -)$.

- $\mathbb{A}\mathrm{gt} = \{\mathbf{v}, \mathbf{r}\}$,
- $Q = \{q_0, q'_0, q_d\} \cup \{q_{\mathbf{v}}, q'_{\mathbf{v}}, q_{\mathbf{r}}, q'_{\mathbf{r}}\} \cup Q_1 \cup Q_2 \cup Q_3 \cup Q_4 \cup Q_5$,
- $\Pi = \{\mathsf{x_i} \mid i = 1, \ldots, n\} \cup \{\mathsf{notx_i} \mid i = 1, \ldots, n\} \cup \{\mathsf{win_v}, \mathsf{win_r}, \mathsf{start}, \mathsf{d}, \mathsf{i_r}, \mathsf{i_v}\}$,
- $\pi(q_0) = \{\mathsf{start}\}$, $\pi(q'_{\mathbf{v}}) = \{i_{\mathbf{v}}\}$, $\pi(q'_0) = \pi(q'_{\mathbf{r}}) = \{i_{\mathbf{r}}\}$, $\pi(q_{i\top}) = \{\mathsf{x_i}\}$ for all $q_{i\top} \in Q_1$, $\pi(q_{i\bot}) = \{\mathsf{notx_i}\}$ for all $q_{i\top} \in Q_1$, $\pi(q_{x_i\top}) = \{\mathsf{x_i}\}$ for all $q_{x_i\top} \in Q_4$, $\pi(q_{x_i\bot}) = \{\mathsf{notx_i}\}$ for all $q_{x_i\top} \in Q_5$, $\pi(q_{\neg x_i\bot}) = \{\mathsf{x_i}\}$ for all $q_{\neg x_i\top} \in Q_4$, $\pi(q_{\neg x_i\top}) = \{\mathsf{notx_i}\}$ for all $q_{\neg x_i\top} \in Q_5$, $\pi(q_\top) = \{\mathsf{win_v}\}$, $\pi(q_d) = \{\mathsf{d}\}$ and $\pi(q_\bot) = \{\mathsf{win_r}\}$.
- $Act = \{L, R, \top, \bot, -\}$
- the function d is defined as
  - $d_{\mathbf{v}}(q_0) = d_{\mathbf{r}}(q_0) = \{\top, \bot\}$.
  - $d_{\mathbf{v}}(q_i) = \{\top, \bot\}$ and $d_{\mathbf{r}}(q_i) = \{-\}$ for $x_i \in X_{\mathbf{v}} \cup \{q_{\mathbf{v}}\}$.
  - $d_{\mathbf{r}}(q_i) = \{\top, \bot\}$ and $d_{\mathbf{v}}(q_i) = \{-\}$ for $x_i \in X_{\mathbf{r}} \cup \{q_{\mathbf{r}}\}$
  - $d_{\mathbf{v}}(q_\iota) = \{L, R\}$ and $d_{\mathbf{r}}(q_i) = \{-\}$ for all $\iota \in \mathsf{ind}(\phi)$ with $\mathsf{sf}(\iota) = \psi_1 \vee \psi_2$ and $\mathsf{sf}(\iota)$ not a literal
  - $d_{\mathbf{r}}(q_\iota) = \{L, R\}$ and $d_{\mathbf{v}}(q_i) = \{-\}$ for all $\iota \in \mathsf{ind}(\phi)$ with $\mathsf{sf}(\iota) = \psi_1 \wedge \psi_2$ and $\mathsf{sf}(\iota)$ not a literal
  - $d_{\mathbf{v}}(q) = d_{\mathbf{r}}(q) = \{-\}$ for all other states q.
- and o is defined as:
  - $o(q_0, (\top, \top)) = q_d$, $o(q_0, (\bot, \top)) = q'_0$, $o(q_0, (\alpha_1, \alpha_2)) = q_{\mathbf{v}}$ for $(\alpha_1, \alpha_2) \in \{(\top, \bot), (\bot, \bot)\}$, and $o(q'_0, (-, -)) = q_{\mathbf{v}}$.
  - $o(q_{\mathbf{v}}, (\top, -)) = q'_{\mathbf{v}}$, $o(q_{\mathbf{v}}, (\bot, -)) = q_{\mathbf{r}}$ and $o(q'_{\mathbf{v}}, (-, -)) = q_{\mathbf{r}}$
  - $o(q_{\mathbf{r}}, (-, \top)) = q'_{\mathbf{r}}$, $o(q_{\mathbf{r}}, (-, \bot)) = q_1$ and $o(q'_{\mathbf{r}}, (-, -)) = q_1$
  - $o(q, (-, -)) = q$ for $q \in \{q_d, q_\top, q_\bot\}$.
  - $o(q_i, (v, -)) = q_{iv}$ where $v \in \{\top, \bot\}$ and $x_i \in X_{\mathbf{v}}$
  - $o(q_i, (-, v)) = q_{iv}$ where $v \in \{\top, \bot\}$ and $x_i \in X_{\mathbf{r}}$
  - $o(q_\iota, (x, -)) = q_{\iota x}$ where $x \in \{L, R\}$, $\iota \in \mathsf{ind}(\phi)$ and $\mathsf{sf}(\iota) = \psi_1 \vee \psi_2$.
  - $o(q_\iota, (-, x)) = q_{\iota x}$ where $x \in \{L, R\}$, $\iota \in \mathsf{ind}(\phi)$ and $\mathsf{sf}(\iota) = \psi_1 \wedge \psi_2$.

- $o(q_\iota, (y, -)) = q_{\iota y}$ where $y \in \{\top, \bot\}$ and $\mathsf{sf}(\iota) = x \in X_{\mathbf{v}}$.
- $o(q_\iota, (y, -)) = q_{\iota z}$ where $y, z \in \{\top, \bot\}$, $y \neq z$ and $\mathsf{sf}(\iota) = \neg x$ with $x \in X_{\mathbf{v}}$.
- $o(q_\iota, (-, y)) = q_{\iota y}$ where $y \in \{\top, \bot\}$ and $\mathsf{sf}(\iota) = x \in X_{\mathbf{r}}$.
- $o(q_\iota, (-, y)) = q_{\iota z}$ where $y, z \in \{\top, \bot\}$, $y \neq z$ and $\mathsf{sf}(\iota) = \neg x$ with $x \in X_{\mathbf{r}}$.
- $o(q_{iv}, (-, -)) = q_{i+1}$ for $i = 1, \ldots, n-1$ and $v \in \{\top, \bot\}$
- $o(q_{nv}, (-, -)) = q_\phi$ for $v \in \{\top, \bot\}$
- $o(q_{\iota\top}, (-, -)) = q_\top$ for $\mathsf{sf}(\iota)$ a literal.
- $o(q_{\iota\bot}, (-, -)) = q_\bot$ for $\mathsf{sf}(\iota)$ a literal.

Each state of $\mathfrak{M}^\phi$ has at most two outgoing transitions, with the exception of state q_0 which has four. Hence, the number of transitions is polynomial in the number of states. In the next section we describe how to ensure that only strategies of the players are taken into consideration which correspond to truth assignments.

B.1.2. Strategies and assignments

If a path in the model goes through a state labelled $\mathsf{x_i}$ and $\mathsf{notx_i}$ this can be interpreted as setting variable x_i true and false, respectively. We observe that the states which have as children an $\mathsf{x_i}$-state or a $\mathsf{notx_i}$-state the transitions to a successor is determined by a single player only. Thus, a strategy of a player completely determines which $\mathsf{x_i}$-states and $\mathsf{notx_i}$-states, where x_i is a variable of the very player, are visited. It may happen that a path contains an $\mathsf{x_i}$-state as well as a $\mathsf{notx_i}$-state, by performing contrary actions in a state q_i of the value-choice section and in q_ι with $\mathsf{sf}(\iota) = x_i$ in the literal section. If this happens we call the responsible strategy of the player *inconsistent* as it does not encode a truth assignment of the variables that the player controls. Thus, it has to be ensured that choices are made consistently: the same variable x is always assigned true, or always assigned false. For this purpose the following *consistency constraints*—temporal formulae—are introduced for each $i \in \{1, \ldots, n\}$:

$$\mathsf{T}_i \equiv \Box\neg\mathsf{x_i} \vee \Box\neg\mathsf{notx_i}.$$

It is easy to see that if T_i is true along a path then this path—or the associated strategy of the player that controls x_i—represents a truth assignment of x_i. We set

$$\mathsf{T}_{\mathbf{v}} \equiv \Box\neg i_{\mathbf{v}} \wedge \bigwedge_{i=1}^{m} \mathsf{T}_i$$

$$\mathsf{T}_{\mathbf{r}} \equiv \Box\neg i_{\mathbf{r}} \wedge \bigwedge_{i=m+1}^{n} \mathsf{T}_i$$

where the propositions $i_{\mathbf{r}}$ and $i_{\mathbf{v}}$ are used to indicate that the current strategy of the refuter and verifier, respectively, are *invalid*. The latter is needed to prevent specific strategy profiles to constitute a Nash equilibrium. We call a strategy of the verifier $\mathbf{v}$ (resp. refuter $\mathbf{r}$) *consistent* if for all strategies of the refuter (resp. the verifier) the induced outcome path satisfies $\mathsf{T}_{\mathbf{v}}$ (resp. $\mathsf{T}_{\mathbf{r}}$). In the case of consistent choices we observe that the formula structure section together with the sections of literals implement the game theoretical semantics of Boolean formula [39].

Next we recall from [22] (with small modifications) the following lemma which says that ϕ is satisfiable iff there is a consistent strategy of player $\mathbf{v}$ such that for all consistent strategies of player $\mathbf{r}$—again, consistent means that truth values are assigned consistently to variables and that it is not invalid—such that eventually $\mathsf{win}_{\mathbf{v}}$ holds. We emphasize that our construction does not assume perfect-recall strategies due to the fact the we are only considering one quantifier alternation. Also, note that if $q_\bot$ or $q_\top$ is reached, each agent can switch from a consistent strategy to an inconsistent one without changing the outcome paths with the possible exception of the transitions between $q_{\mathbf{v}}$ and q_1.

Lemma 16. *Let a* ***QSAT***$_2$*-formula $\phi \equiv \exists\{x_1, \ldots, x_m\}\forall\{x_{m+1}, \ldots, x_n\}\varphi(x_1, \ldots, x_n)$ in negated normal form be given. The model $\mathfrak{M}^\phi$ can be constructed in polynomial time. We have that ϕ is satisfiable iff there is a strategy $s_{\mathbf{v}}$ of the verifier $\mathbf{v}$ with $s_{\mathbf{v}}(q_0) = \bot$ such that for all strategies $s_{\mathbf{r}}$ of the refuter $\mathbf{r}$ it holds that $out_{\mathfrak{M}^\phi}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} \mathsf{T}_{\mathbf{v}} \wedge (\mathsf{T}_{\mathbf{r}} \rightarrow \Diamond\mathsf{win}_{\mathbf{v}})$.*

Proof. "⇒": Suppose ϕ is true and let $s_{\mathbf{v}}$ be the strategy induced by a truth assignment v of $\{x_1, \ldots, x_m\}$ witnessing the truth of ϕ (i.e. $\varphi[v]$ is valid) and by the simulation of the game theoretic semantics which witnesses the truth of $\varphi[v]$. Moreover, let $s_{\mathbf{v}}(q_0) = \bot$ (this is needed to avoid that the refuter can reach the state q_d) and let $s_{\mathbf{r}}$ be any consistent strategy of $\mathbf{r}$. Note that the strategy $(s_{\mathbf{v}}, s_{\mathbf{r}})$ induces a truth assignment $v_1 \circ v_2$ of $X_{\mathbf{v}} \cup X_{\mathbf{r}}$. Then, $\Diamond\mathsf{win}_{\mathbf{v}}$ must be true on $out_{\mathfrak{M}^\phi}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}}))$ otherwise the refuter had a strategy $s_{\mathbf{r}}$ which induces a truth assignment v with $\varphi[v_1 \circ v]$ is false. Which would yield a contradiction.

"⇐": Let $s_{\mathbf{v}}$ be a strategy of the verifier $\mathbf{v}$ with $s_{\mathbf{v}}(q_0) = \bot$ such that for all strategies $s_{\mathbf{r}}$ of the refuter $\mathbf{r}$ it holds that $out_{\mathfrak{M}^\phi}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} \mathsf{T}_{\mathbf{v}} \wedge (\mathsf{T}_{\mathbf{r}} \rightarrow \Diamond\mathsf{win}_{\mathbf{v}})$. $s_{\mathbf{v}}$ induces a truth assignment v. For any consistent strategy $s_{\mathbf{r}}$ we have that $\Diamond\mathsf{win}_{\mathbf{v}}$ is true. It is straight-forward to check that the truth assignment v induced by $s_{\mathbf{v}}$ encodes a truth assignment witnessing the truth of ϕ due to the encoded game theoretic semantics in the model. That is, $\varphi[v]$ is valid and thus ϕ is true. □

B.1.3. Preferences and Nash equilibria

Lemma 16 showed that the satisfiability problem of a given **QSAT**$_2$-formula ϕ can be reduced to a strategy existence and model checking problem of a rather simple formula in $\mathfrak{M}^\phi$. What we need for the reduction to the weak implementability problem is that a winning strategy of the verifier is part of a Nash equilibrium iff the formula ϕ is satisfiable. For this purpose, we define the following preference lists for player $\mathbf{v}$ and $\mathbf{r}$, respectively:

$$\begin{aligned}\gamma_{\mathbf{v}} = (&((\bigcirc \mathsf{d}, 4),\\ &((\Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{v}}), 3),\\ &((\neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg\mathsf{T}_{\mathbf{r}} \wedge \mathsf{T}_{\mathbf{v}}), 2),\\ &((\neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}} \wedge \neg\mathsf{T}_{\mathbf{v}}), 1),\\ &(\top, 0)),\end{aligned} \qquad \begin{aligned}\gamma_{\mathbf{r}} = (&((\bigcirc \mathsf{d}, 4),\\ &((\neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}} \wedge \mathsf{T}_{\mathbf{v}}), 3),\\ &((\neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg\mathsf{T}_{\mathbf{r}} \wedge \neg\mathsf{T}_{\mathbf{v}}), 2),\\ &((\Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}}), 1),\\ &(\top, 0)).\end{aligned}$$

Now we can relate the satisfiability of ϕ to the existence of Nash equilibria which contains a winning strategy of the verifier according to Lemma 16.

Proposition 17 *(Nash equilibria in $\mathfrak{M}^\phi$). Let ϕ be a **QSAT**$_2$-formula in nnf and $(\gamma_{\mathbf{v}}, \gamma_{\mathbf{r}})$ be the preference profile defined above.*

(a) *We have that $(s_d, s'_d) \in \mathcal{NE}(\mathfrak{M}^\phi, q_0, (\gamma_{\mathbf{v}}, \gamma_{\mathbf{r}}))$ for any two strategies s_d and s'_d with $s_d(q_0) = s'_d(q_0) = \top$.*
(b) *ϕ is satisfiable iff there is an $s \in \mathcal{NE}(\mathfrak{M}^\phi, q_0, (\gamma_{\mathbf{v}}, \gamma_{\mathbf{r}}))$ with $out_{\mathfrak{M}^\phi}(q_0, s) \models^{\mathbf{LTL}} \Diamond \mathsf{win}_{\mathbf{v}}$.*
(c) *Let $\gamma'_{\mathbf{v}}$ and $\gamma'_{\mathbf{r}}$ equal $\gamma_{\mathbf{v}}$ and $\gamma_{\mathbf{r}}$ but with the first list entry $(\bigcirc \mathsf{d}, 4)$ being removed, respectively. Then, it holds that ϕ is satisfiable iff $\mathcal{NE}(\mathfrak{M}^\phi, q_0, (\gamma'_{\mathbf{v}}, \gamma'_{\mathbf{r}})) \neq \emptyset$.*

Proof.

(a) The strategy profile (s_d, s'_d) yields the path $q_0 q_d^\omega$. For that path the payoff for both players is four. No player can deviate to improve its payoff. It is a Nash equilibrium.
(b) "$\Rightarrow$": If ϕ holds then, according to Lemma 16, there is a strategy $s_{\mathbf{v}}$ such that $s_{\mathbf{v}}(q_0) = \bot$ and for all strategies $s_{\mathbf{r}}$ it holds that $out_{\mathfrak{M}^\phi}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models \mathsf{T}_{\mathbf{v}} \wedge (\mathsf{T}_{\mathbf{r}} \rightarrow \Diamond \mathsf{win}_{\mathbf{v}})$ ($\star$). We claim that the profile $(s_{\mathbf{v}}, s'_{\mathbf{r}})$ for any consistent strategy $s'_{\mathbf{r}}$ of the refuter is a Nash equilibrium. First, observe that $s_{\mathbf{r}}(q_0) = \bot$, otherwise $\mathsf{T}_{\mathbf{r}}$ could not be true on the outcome path. Second, note that such a strategy $s'_{\mathbf{r}}$ must exist as $\mathbf{r}$ always has a consistent strategy. Given these strategies players $\mathbf{v}$ and $\mathbf{r}$ get a payoff of 3 and 1, respectively. As $s_{\mathbf{r}}(q_0) = s_{\mathbf{v}}(q_0) = \bot$ no player can deviate to reach state q_d. Moreover, player $\mathbf{r}$ cannot deviate to a consistent strategy making $\neg\Diamond \mathsf{win}_{\mathbf{v}}$ true by Lemma 16. This shows that no player can unilaterally deviate to increase its payoff.
"$\Leftarrow$": Suppose ϕ is false. That is,

($\star$) for each truth assignment v_1 of $\{x_1, \ldots, x_m\}$ there is a truth assignment v_2 of $\{x_{m+1}, \ldots, x_n\}$ such that $\varphi[v_1 \circ v_2]$ is false.

We consider a strategy profile $s = (s_{\mathbf{v}}, s_{\mathbf{r}})$ with $out_{\mathfrak{M}^\phi}(q_1, s) \models^{\mathbf{LTL}} \Diamond \mathsf{win}_{\mathbf{v}}$ and show that it cannot be a Nash equilibrium. For the sake of contradiction, suppose s were a Nash equilibrium. Then, first, it must be the case that $\mathsf{T}_{\mathbf{v}}$ is true on the path; otherwise, player $\mathbf{v}$ can increase its utility by deviating to a consistent strategy that still satisfies $\Diamond \mathsf{win}_{\mathbf{v}}$ (note that the strategy of $\mathbf{r}$ is fixed and memoryless). Second, we show that player $\mathbf{r}$ can always deviate to increase its payoff. If T_{r} holds then $\mathbf{r}$ gets a payoff of 1. However, by ($\star$) the refuter can deviate to a consistent strategy such that the resulting path satisfies $\neg\Diamond \mathsf{win}_{\mathbf{v}}$. By doing so it can increase its payoff to 3. On the other hand, if T_{r} does not hold, then the refuter gets a payoff of 0 and can again deviate to increase its payoff by ($\star$), i.e. by deviating to a consistent strategy that satisfies $\neg\Diamond \mathsf{win}_{\mathbf{v}}$.
(c) We consider the modified preference list. If ϕ is satisfiable then $\mathcal{NE}(\mathfrak{M}^\phi, q_0, (\gamma'_{\mathbf{v}}, \gamma'_{\mathbf{r}})) \neq \emptyset$ by (b). Now suppose ϕ is false. That is, ($\star$) holds. We consider a strategy profile $s = (s_{\mathbf{v}}, s_{\mathbf{r}})$ and show that it cannot be a Nash equilibrium. Firstly, suppose $out_{\mathfrak{M}^\phi}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models \Diamond \mathsf{win}_{\mathbf{v}}$. Then, it has already been shown in (b) that s cannot be a Nash equilibrium. Thus, we consider the case that $\lambda := out_{\mathfrak{M}^\phi}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models \neg\Diamond \mathsf{win}_{\mathbf{v}}$. We distinguish the four cases which result from players playing consistent or inconsistent strategies. (i) Suppose $\mathsf{T}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}}$ holds on path λ. Then, $\mathbf{v}$ is better off by playing an inconsistent strategy resulting in a payoff of 1 rather than 0. (ii) If $\mathsf{T}_{\mathbf{v}} \wedge \neg\mathsf{T}_{\mathbf{r}}$ is true then $\mathbf{r}$ is better of playing a consistent strategy which remains to satisfy $\neg\Diamond \mathsf{win}_{\mathbf{v}}$. This is possible by ($\star$). (iii) $\neg\mathsf{T}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}}$ is true on λ. Then, player $\mathbf{r}$ can deviate to an inconsistent strategy that still makes $\neg\Diamond \mathsf{win}_{\mathbf{v}}$ true (this can be achieved by playing $\top$ in state q_0). This increases the player's payoff from 0 to 2. (iv) Suppose the path satisfies $\neg\mathsf{T}_{\mathbf{v}} \wedge \neg\mathsf{T}_{\mathbf{r}}$. Then, player $\mathbf{v}$ is better off to deviate to a consistent strategy. This shows that there cannot be any Nash equilibrium. □

Theorem 18 *(Hardness of weak implementation: verification). Let $(\mathcal{I}, q)$ be a pointed NIS and N a normative system included in $\mathcal{I}$. The problem whether $\mathsf{N} \in \mathsf{WI}_{\mathcal{NE}}(\mathcal{I}, q, f)$ is $\mathbf{\Sigma_2^P}$-hard in the size of $\mathfrak{M}$, N and $\mathcal{P}refs$.*

Proof. $\mathbf{\Sigma_2^P}$-hardness is shown by a reduction to the weak implementability problem of $f((\gamma_{\mathbf{v}}, \gamma_{\mathbf{r}})) = \Diamond \mathsf{win}_{\mathbf{v}}$. Given a **QSAT**$_2$-formula ϕ in nnf we construct the model $\mathfrak{M}^{\phi}$, which can be done in polynomial time according to Lemma 16, and define $\mathfrak{I} = (\mathfrak{M}^{\phi}, \{(\gamma_{\mathbf{v}}, \gamma_{\mathbf{r}})\}, \{N_{\epsilon}\})$. Now, by Proposition 17(b) we have that: ϕ is satisfiable iff $\exists s \in \mathcal{NE}(\mathfrak{M}^{\phi}, q_0, (\gamma_{\mathbf{v}}, \gamma_{\mathbf{r}}))$ such that $out(q_0, s) \models \Diamond \mathsf{win}_{\mathbf{v}}$. This equivalence, however, is equivalent to $\mathsf{N}_{\epsilon} \in \mathsf{WI}_{\mathcal{NE}}(\mathfrak{I}, q_0, f)$. □

B.2. Hardness of strong implementability: Theorem 5

The strong implementation problem requires checking two parts: that the set of Nash equilibria is non-empty and that all Nash equilibria satisfy specific properties. We show that the latter one gives rise to $\mathbf{\Pi_2^P}$-hardness. In order to be able to use our model $\mathfrak{M}^{\phi}$ for a reduction of $\forall$**QSAT**$_2$, we need to switch the roles of the verifier and refuter. That is, the refuter controls variables in $\{x_1, \ldots, x_m\}$ and the verifier in $\{x_{m+1}, \ldots, x_n\}$. This revised model denoted by $\widehat{\mathfrak{M}}^{\phi}$ and will later also be used for the problem of the existence of an appropriate normative system. This reduction requires an additional labelling of some states, modelling the guessing of a normative system.

Definition 27 *(The model $\widehat{\mathfrak{M}}^{\phi}$).* Given a $\forall$**QSAT**$_2$-formula $\phi \equiv \forall\{x_1, \ldots, x_m\}\exists\{x_{m+1}, \ldots, x_n\}\varphi(x_1, \ldots, x_n)$ in nnf we define the CGS $\widehat{\mathfrak{M}}^{\phi}$ analogously to $\mathfrak{M}^{\phi}$ but $X_{\mathbf{r}} = \{x_1, \ldots, x_m\}$ and $X_{\mathbf{v}} = \{x_{m+1}, \ldots, x_n\}$. We further label q_0 and each state reachable in an even number of transitions from q_0 with a new proposition t.

The labelling t will later be used for ensuring that the structure of $\widehat{\mathfrak{M}}^{\phi}$ is not affected by a norm-based update, more precisely that no transitions are regimented making for instance the winning state of the verifier unreachable. The proof of the following lemma is done analogously to the one of Lemma 16.

Lemma 19. *Given a $\forall$**QSAT**$_2$-formula $\phi \equiv \forall\{x_1, \ldots, x_m\}\exists\{x_{m+1}, \ldots, x_n\}\varphi(x_1, \ldots, x_n)$ in nnf we have that ϕ is satisfiable iff for all strategies $s_{\mathbf{r}}$ of refuter $\mathbf{r}$ with $s_{\mathbf{r}}(q_0) = \bot$ there is a strategy $s_{\mathbf{v}}$ of verifier $\mathbf{v}$ such that $out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} \mathsf{T}_{\mathbf{r}} \rightarrow (\mathsf{T}_{\mathbf{v}} \wedge \Diamond \mathsf{win}_{\mathbf{v}})$.*

Proof. [Sketch] "⇒": Suppose ϕ is true. For each truth assignment v of $X_{\mathbf{r}}$ let w^v denote a truth assignment of $X_{\mathbf{v}}$ such that $\varphi[v \circ w^v]$ is valid. Let $s_{\mathbf{r}}$ denote the strategy induced by v and let $s_{\mathbf{v}}^{s_{\mathbf{r}}}$ be a consistent strategy induced by w^v which witnesses the truth of ϕ. Thus, $\Diamond \mathsf{win}_{\mathbf{v}}$ must be true on $out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, (s_{\mathbf{v}}^{s_{\mathbf{r}}}, s_{\mathbf{r}}))$ otherwise the refuter had a consistent strategy for which there is no consistent strategy $s_{\mathbf{v}}^{s_{\mathbf{r}}}$ such that $out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, (s_{\mathbf{v}}^{s_{\mathbf{r}}}, s_{\mathbf{r}})) \not\models^{\mathbf{LTL}} \Diamond \mathsf{win}_{\mathbf{v}}$. This would imply that there is an v such that for all w^v, $\varphi[v \circ w^v]$ is false. Contradiction.

"⇐": Let $s_{\mathbf{v}}^{s_{\mathbf{r}}}$ be a consistent strategy of the verifier $\mathbf{v}$ which witnesses the truth of the formulae when the refuter plays the consistent strategy $s_{\mathbf{r}}$, i.e. $out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, (s_{\mathbf{v}}^{s_{\mathbf{r}}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} \Diamond \mathsf{win}_{\mathbf{v}}$. The strategies induce the truth assignments v and w^v, respectively (using the same notation as above). It is straight-forward to check that $\varphi[v \circ w^v]$ is true. □

The next lemma shows that we can use the previous result to reduce the truth of a $\forall$**QSAT**$_2$ instance to a property over all Nash equilibria in $\widehat{\mathfrak{M}}_{\phi}$ using a slightly modified variant of the preference lists of player $\mathbf{v}$ used before. We define

$$\begin{aligned}\widehat{\gamma}_{\mathbf{v}} = (&((\bigcirc \mathsf{d}, 5),\\ &((\Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{v}}), 4),\\ &((\neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg \mathsf{T}_{\mathbf{r}} \wedge \mathsf{T}_{\mathbf{v}}), 3),\\ &((\neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}} \wedge \mathsf{T}_{\mathbf{v}}), 2),\\ &((\neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}} \wedge \neg \mathsf{T}_{\mathbf{v}}), 1),\\ &(\top, 0)),\end{aligned}$$

and $\widehat{\gamma}_{\mathbf{r}} = \gamma_{\mathbf{r}}$.

Proposition 20 *(Nash equilibria in $\widehat{\mathfrak{M}}^{\phi}$). Let ϕ be a $\forall$**QSAT**$_2$-formula in nnf.*

(a) We have that $(s_d, s'_d) \in \mathcal{NE}(\widehat{\mathfrak{M}}^{\phi}, q_0, (\widehat{\gamma}_{\mathbf{v}}, \widehat{\gamma}_{\mathbf{r}}))$ for any strategies s_d and s'_d with $s_d(q_0) = s'_d(q_0) = \top$.
(b) ϕ is satisfiable iff for all $s \in \mathcal{NE}(\widehat{\mathfrak{M}}^{\phi}, q_0, (\widehat{\gamma}_{\mathbf{v}}, \widehat{\gamma}_{\mathbf{r}}))$ it holds that $out(q_0, s) \models^{\mathbf{LTL}} \Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}$.

Proof.

(a) Analogously to Proposition 17(a).
(b) "⇒": If ϕ holds then, according to Lemma 19, for all strategies $s_{\mathbf{r}}$ with $s_{\mathbf{r}}(q_0) = \bot$ there is a strategy $s_{\mathbf{v}}$ such that $out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}}))) \models \mathsf{T}_{\mathbf{r}} \rightarrow (\mathsf{T}_{\mathbf{v}} \wedge \Diamond \mathsf{win}_{\mathbf{v}})$ (⋆). Now, suppose there was a Nash equilibrium $s = (s_{\mathbf{v}}, s_{\mathbf{r}})$ with $out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models \neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg \bigcirc \mathsf{d}$. We consider the following cases. First, assume that $out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models \mathsf{T}_{\mathbf{r}}$.

Then, by ($\star$) the verifier could deviate from $s_{\mathbf{v}}$ to obtain a path satisfying $\mathsf{T}_{\mathbf{v}} \wedge \Diamond \mathsf{win}_{\mathbf{v}}$. This shows that $(s_{\mathbf{v}}, s_{\mathbf{r}})$ is not a Nash equilibrium. Second, assume that $out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models \neg \mathsf{T}_{\mathbf{r}} \wedge \mathsf{T}_{\mathbf{v}}$. In that case, the refuter would be better off playing a consistent strategy. Thirdly, assume that $out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models \neg \mathsf{T}_{\mathbf{r}} \wedge \neg \mathsf{T}_{\mathbf{v}}$. In that case the verifier would be better off switching to a consistent strategy. Also note that not both players can play $\top$ in q_0 as $\neg \bigcirc \mathsf{d}$ holds. This shows that for all $s \in \mathcal{NE}(\widehat{\mathfrak{M}}^{\phi}, q_0, (\widehat{\gamma}_{\mathbf{v}}, \widehat{\gamma}_{\mathbf{r}}))$, $out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, s) \models^{\mathbf{LTL}} \Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}$.

"$\Leftarrow$": Suppose ϕ is false. We need to show that there is a Nash equilibrium s such that $out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, s) \models^{\mathbf{LTL}} \neg \Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg \bigcirc \mathsf{d}$. By Lemma 19 there is a strategy $s_{\mathbf{r}}$ with $s_{\mathbf{r}}(q_0) = \bot$ such that for all strategies $s_{\mathbf{v}}$ we have $out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} \mathsf{T}_{\mathbf{r}} \wedge (\mathsf{T}_{\mathbf{v}} \rightarrow \neg \Diamond \mathsf{win}_{\mathbf{v}})$ (by contraposition). Now, let $s'_{\mathbf{v}}$ be any strategy such that $out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, (s'_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} \mathsf{T}_{\mathbf{v}}$ and $s'_{\mathbf{v}}(q_0) = \bot$. We show that $(s_{\mathbf{r}}, s'_{\mathbf{v}}) \in \mathcal{NE}(\widehat{\mathfrak{M}}^{\phi}, q_0, (\widehat{\gamma}_{\mathbf{v}}, \widehat{\gamma}_{\mathbf{r}}))$. By Lemma 19, the verifier cannot change its strategy to ensure $\Diamond \mathsf{win}_{\mathbf{v}}$. Moreover, as $s_{\mathbf{r}}(q_0) = \bot$ the verifier $\mathbf{v}$ cannot deviate to increase its payoff. Analogously, as $s'_{\mathbf{v}}(q_0) = \bot$, the refuter cannot deviate from $s_{\mathbf{r}}$ to increase its payoff. This shows that $(s_{\mathbf{r}}, s'_{\mathbf{v}})$ is indeed a Nash equilibrium. Thus we have that there is an $s \in \mathcal{NE}(\widehat{\mathfrak{M}}^{\phi}, q_0, (\widehat{\gamma}_{\mathbf{v}}, \widehat{\gamma}_{\mathbf{r}}))$ with $out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, s) \models^{\mathbf{LTL}} \neg \Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg \bigcirc \mathsf{d}$. □

Theorem 21 *(Hardness of strong implementation: verification). Let $(\mathfrak{I}, q)$ be a pointed NIS and N a normative system included in $\mathfrak{I}$. The problem whether $\mathsf{N} \in \mathsf{SI}_{\mathcal{NE}}(\mathfrak{I}, q, f)$ is $\mathbf{\Sigma}_2^{\mathbf{P}}$-hard as well as $\mathbf{\Pi}_2^{\mathbf{P}}$-hard in the size of $\mathfrak{M}$, N and $\mathcal{P}refs$.*

Proof. $\mathbf{\Sigma}_2^{\mathbf{P}}$-hardness. By Lemma 16 and Proposition 17(c) we can construct a model $\mathfrak{M}^{\phi}$ from a **QSAT**$_2$-formula ϕ in nnf in polynomial time such that: ϕ is satisfiable iff $\mathcal{NE}(\mathfrak{M}^{\phi}, q_0, (\widehat{\gamma}_{\mathbf{v}}, \widehat{\gamma}_{\mathbf{r}})) \neq \emptyset$. This is equivalent to

$$\mathcal{NE}(\mathfrak{M}^{\phi}, q_0, (\widehat{\gamma}_{\mathbf{v}}, \widehat{\gamma}_{\mathbf{r}})) \neq \emptyset \text{ and } \forall s \in \mathcal{NE}(\mathfrak{M}^{\phi}, q_0, (\widehat{\gamma}_{\mathbf{v}}, \widehat{\gamma}_{\mathbf{r}})) : out_{\mathfrak{M}^{\phi}}(q_0, s) \models^{\mathbf{LTL}} \top \quad \text{iff} \quad \mathsf{N}_{\epsilon} \in \mathsf{SI}_{\mathcal{NE}}(\mathfrak{I}, q_0, f)$$

for $f((\widehat{\gamma}_{\mathbf{v}}, \widehat{\gamma}_{\mathbf{r}})) = \top$.

$\mathbf{\Pi}_2^{\mathbf{P}}$-hardness. We reduce $\forall$**QSAT**$_2$ to the strong implementation problem for $f((\widehat{\gamma}_{\mathbf{v}}, \widehat{\gamma}_{\mathbf{r}})) = \Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}$. By Lemma 19 and Proposition 20 we can construct a model $\widehat{\mathfrak{M}}^{\phi}$ from a $\forall$**QSAT**$_2$-formula ϕ in polynomial time such that:

ϕ is satisfiable

iff $\forall s \in \mathcal{NE}(\widehat{\mathfrak{M}}^{\phi}, q_0, (\widehat{\gamma}_{\mathbf{v}}, \widehat{\gamma}_{\mathbf{r}})) : out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, s) \models^{\mathbf{LTL}} \Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}$ (Proposition 20(b))

iff $\forall s \in \mathcal{NE}(\widehat{\mathfrak{M}}^{\phi}, q_0, (\widehat{\gamma}_{\mathbf{v}}, \widehat{\gamma}_{\mathbf{r}})) : out_{\widehat{\mathfrak{M}}^{\phi}}(q_0, s) \models^{\mathbf{LTL}} \Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}$ and $\mathcal{NE}(\widehat{\mathfrak{M}}^{\phi}, q_0, (\widehat{\gamma}_{\mathbf{v}}, \widehat{\gamma}_{\mathbf{r}})) \neq \emptyset$ (Proposition 20(a))

iff $\mathsf{N}_{\epsilon} \in \mathsf{SI}_{\mathcal{NE}}(\mathfrak{I}, q_0, f)$ □

Appendix C. Proofs of implementation problems: existence

In the following we consider the hardness proof of the existence problem stated in Proposition 8. The proof that the problem is $\mathbf{\Sigma}_3^{\mathbf{P}}$-hard requires a further technical sophistication. The basic idea is that the normative system is used to simulate the first existential quantification in a **QSAT**$_3$-formula. Firstly, we show how this can be achieved by using sanctioning norms and afterwards by using regimenting norms.

C.1. Sanctioning and regimentation norms

For the hardness part we reduce **QSAT**$_3$. Therefore, we consider the **QSAT**$_3$-formula

$$\phi \equiv \exists\{x_1, \ldots, x_r\} \forall\{x_{r+1}, \ldots, x_{r+m}\} \exists\{x_{r+m+1}, \ldots, x_n\} \varphi(x_1, \ldots, x_n)$$

in nnf and show that ϕ is true iff $\mathsf{SI}_{\mathcal{NE}}(\mathfrak{I}, q, f) = \emptyset$ for appropriate $\mathfrak{I}$, q, and f. The idea of the reduction extends the reduction given in the previous section. Essentially, we use the construction $\widehat{\mathfrak{M}}^{\widehat{\phi}}$ from the previous section where $\widehat{\phi}$ is a modified version of the **QSAT**$_3$-formula ϕ. The refuter $\mathbf{r}$ additionally controls the literal states referring to the new variables in $\{x_1, \ldots, x_r\}$. In order to use our previous construction, we define $\widehat{\phi}$ as the formula obtained from ϕ as follows:

$$\widehat{\phi} \equiv \forall\{x_1, \ldots, x_r, x_{r+1}, \ldots, x_{r+m}\} \exists\{x_{r+m+1}, \ldots, x_n\} \varphi(x_1, \ldots, x_n)$$

That is, the first existentially quantified variable are moved to the universal part controlled by the refuter. A sanctioning norm is used to define a truth assignment of the variables $x_1, \ldots, x_r$. For this purpose, a sanctioning norm labels states q'_0 and $q_{\mathbf{v}}$ with new propositions representing the truth assignment. For illustration assume that the sanctioning norm should encode a truth assignment which assigns true to $x_1, \ldots, x_g$ and false to $x_{g+1}, \ldots, x_r$ with $1 \leq g \leq r$. This can be modelled by the normative system:

$$\mathsf{N} = \{(\{\mathsf{start}\}, \{(\star_1, \star_2) \mid \star_1, \star_2 \in \{\top, \bot\}, \text{not } \star_1 = \star_2 = \top\}, \{\mathsf{x}_1, \ldots, \mathsf{x}_g, \mathsf{notx}_{g+1}, \ldots, \mathsf{notx}_r)\}.$$

Then, in the model $\widehat{\mathfrak{M}}^{\widehat{\phi}} \upharpoonright \mathsf{N}$, states q'_0 and $q_{\mathbf{v}}$ are additionally labelled with $\{\mathsf{x}_1, \ldots, \mathsf{x}_g, \mathsf{notx}_{g+1}, \ldots, \mathsf{notx}_r\}$. Given that we consider $\widehat{\phi}$ the consistency constraint $\mathsf{T}_{\mathbf{r}}$ of $\mathbf{r}$ in $\widehat{\mathfrak{M}}^{\widehat{\phi}}$ has the form:

$$\mathsf{T}_{\mathbf{r}} \equiv \Box\neg i_{\mathbf{r}} \wedge \bigwedge_{i=r+1}^{r+m} \mathsf{T}_i \wedge \bigwedge_{i=1}^{r} (\Box\neg \mathsf{x}_i \vee \Box\neg \mathsf{notx}_i).$$

We also define the formula

$$\mathsf{asgn} \equiv \left(\bigcirc \bigwedge_{i=1}^{r} \big((\mathsf{x}_i \vee \mathsf{notx}_i) \wedge \neg(\mathsf{x}_i \wedge \mathsf{notx}_i)\big) \right) \vee \bigcirc \mathsf{d}$$

expressing that the next state is either q_d, or q_0' or $q_{\mathbf{v}}$ each of the two labelled with a representation of a truth assignment for the variables $\{x_1, \ldots, x_r\}$, assuming that the current state is q_0. These formulae suffice if we were only given sanctioning norms. In the presence of regimenting norms, however, it has also to be ensured that regimenting norms do not regiment transitions invalidating the structure of the model. For example, the transitions leading to state $q_\top$ may simply be regimented which makes it impossible for the verifier to win. The idea is to introduce a formula which "forbids" such undesirable normative systems. Therefore, we define the formula

$$\mathsf{tick} = \Box\big((\mathsf{t} \rightarrow \bigcirc\neg\mathsf{t}) \wedge (\neg\mathsf{t} \rightarrow \bigcirc\mathsf{t}) \vee \bigcirc(\mathsf{d} \vee \mathsf{win}_{\mathbf{v}} \vee \mathsf{win}_{\mathbf{r}})\big).$$

It describes that a path does not loop in states other than the already looping states q_d, $q_\top$, and $q_\bot$. Now, we can prove the following result:

Lemma 22. *Let* $\mathsf{N} \in \mathcal{N}_{rs}$ *be a normative system such that for all paths* $\lambda \in \Lambda_{\widehat{\mathfrak{M}}^{\hat{\phi}}{}_{\restriction \mathsf{N}}}(q_0)$ *it holds that* $\lambda \models \mathsf{tick}$. *Then,* N *cannot contain a regimentation norm which is applicable in a state* $Q \backslash \{q_d, q_\bot, q_\top\}$.

Proof. Suppose that an outgoing transition in a state $q \in Q\backslash\{q_d, q_\bot, q_\top\}$ is regimented. Then, there is a loop from q to q. Let λ denote the path from q_0 to q followed by q^ω. Clearly, the formula tick is violated on that path which contradicts the assumption. □

Lemma 23. *Given a* ***QSAT***$_3$*-formula* $\phi \equiv \exists\{x_1, \ldots, x_r\}\forall\{x_{r+1}, \ldots, x_{r+m}\}\exists\{x_{r+m+1}, \ldots, x_n\}\varphi(x_1, \ldots, x_n)$ *in nnf we have that* ϕ *is satisfiable iff there is an* $\mathsf{N} \in \mathcal{N}_{\mathsf{rs}}$ *such that for all strategies* $s_{\mathbf{r}}$ *of* $\mathbf{r}$ *with* $s_{\mathbf{r}}(q_0) = \bot$ *there is a strategy* $s_{\mathbf{v}}$ *of* $\mathbf{v}$ *such that* $out_{\widehat{\mathfrak{M}}^{\hat{\phi}}{}_{\restriction \mathsf{N}}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} (\mathsf{T}_{\mathbf{r}} \rightarrow (\mathsf{T}_{\mathbf{v}} \wedge \Diamond \mathsf{win}_{\mathbf{v}}))$ *and for all paths* $\lambda \in \Lambda_{\widehat{\mathfrak{M}}^{\hat{\phi}}{}_{\restriction \mathsf{N}}}(q_0)$ *we have that* $\lambda \models \mathsf{ass} \wedge \mathsf{tick}$. *Moreover, for the direction "*$\Rightarrow$*" we can always find a normative system in* $\mathcal{N}_{\mathsf{s}}$.

Proof. "$\Rightarrow$": Suppose ϕ holds. Let v be a witnessing truth assignment of the variables $\{x_1, \ldots, x_r\}$. Then, $\phi' \equiv \forall\{x_{r+1}, \ldots, x_{r+m}\}\exists\{x_{r+m+1}, \ldots x_n\}\varphi(x_1, \ldots, x_n)[v]$ is satisfiable. By Lemma 19, we have that for all $s_{\mathbf{r}}$ with $s_{\mathbf{r}}(q_0) = \bot$ there is an $s_{\mathbf{v}}$ such that

$$(\star) \quad out_{\widehat{\mathfrak{M}}^{\phi'}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} \mathsf{T}_{\mathbf{r}} \rightarrow (\mathsf{T}_{\mathbf{v}} \wedge \Diamond \mathsf{win}_{\mathbf{v}}).$$

Now let

$$\mathsf{N} = \{(\{\mathsf{start}\}, \{(\top, \bot), (\bot, \top), (\bot, \bot)\}, \{\mathsf{x}_{i_1}, \ldots, \mathsf{x}_{i_g}, \mathsf{notx}_{i_{g+1}}, \ldots, \mathsf{notx}_{i_r})\}$$

such that $v_{i_1} = \ldots = v_{i_g} = \mathsf{t}$ and $v_{i_{g+1}} = \ldots = v_{i_r} = \mathsf{f}$ where $(i_1, \ldots, i_r)$ is a permutation of $(1, \ldots, r)$. Then, we have that for all paths $\lambda \in \Lambda_{\widehat{\mathfrak{M}}^{\hat{\phi}}{}_{\restriction \mathsf{N}}}(q_0)$, $\pi(\lambda) \models^{\mathbf{LTL}} \mathsf{ass} \wedge \mathsf{tick}$. The claim follows by $(\star)$ applied on $\varphi[v]$ as the normative system essentially fixes the choices of the refuter for all variables $\{x_1, \ldots, x_r\}$; every deviation of these induced truth values of the refuter would result in an inconsistent strategy.

"$\Leftarrow$": Let $\mathsf{N} \in \mathcal{N}_{\mathsf{rs}}$ such that for all strategies $s_{\mathbf{r}}$ with $s_{\mathbf{r}}(q_0) = \bot$ of $\mathbf{r}$ there is a strategy $s_{\mathbf{v}}$ of $\mathbf{v}$ such that $out_{\widehat{\mathfrak{M}}^{\hat{\phi}}{}_{\restriction \mathsf{N}}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} (\mathsf{T}_{\mathbf{r}} \rightarrow (\mathsf{T}_{\mathbf{v}} \wedge \Diamond \mathsf{win}_{\mathbf{v}}))$ and for all paths $\lambda \in \Lambda_{\widehat{\mathfrak{M}}^{\hat{\phi}}{}_{\restriction \mathsf{N}}}(q_0)$ we have that $\lambda \models \mathsf{ass} \wedge \mathsf{tick}$. By Lemma 22 no transitions can be regimented apart from those starting in q_d, $q_\bot$, and $q_\top$. The valuation of q_0' and of $q_{\mathbf{v}}$ induce truth assignments v_1 and v_2 of $\{x_1, \ldots, x_r\}$, respectively. We can choose N in such a way that $v_1 = v_2$. This is the case because for all strategies $s_{\mathbf{r}}$ of $\mathbf{r}$ with $s_{\mathbf{r}}(q_0) = \bot$ there is a strategies $s_{\mathbf{v}}$ of $\mathbf{v}$ such that $out_{\widehat{\mathfrak{M}}^{\hat{\phi}}{}_{\restriction \mathsf{N}}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} (\mathsf{T}_{\mathbf{r}} \rightarrow (\mathsf{T}_{\mathbf{v}} \wedge \Diamond \mathsf{win}_{\mathbf{v}}))$, we also have that $\forall\{x_{r+1}, \ldots x_{r+m}\}\exists\{x_{r+m+1}, \ldots, x_n\}\varphi[v_i]$ is true for $i \in \{1, 2\}$ (cf. Lemma 19). The claim follows. □

Now, we can present our reduction to the implementation problem. Again, we need to slightly modify the players' preference lists:

$$\begin{aligned}
\widehat{\widehat{\gamma}}_{\mathbf{v}} = (&((\bigcirc\mathsf{d} \vee \neg\mathsf{tick} \vee \neg\mathsf{ass}, 5),\\
&((\Diamond\mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{v}}), 4),\\
&((\neg\Diamond\mathsf{win}_{\mathbf{v}} \wedge \neg\mathsf{T}_{\mathbf{r}} \wedge \mathsf{T}_{\mathbf{v}}), 3),\\
&((\neg\Diamond\mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}} \wedge \mathsf{T}_{\mathbf{v}}), 2),
\end{aligned}$$

$$
\begin{aligned}
&((\neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}} \wedge \neg \mathsf{T}_{\mathbf{v}}), 1),\\
&(\top, 0)),\\
\widehat{\widehat{\gamma}}_{\mathbf{r}} = &((\bigcirc \mathsf{d} \vee \neg \mathsf{tick} \vee \neg \mathsf{ass}, 4),\\
&((\neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}} \wedge \mathsf{T}_{\mathbf{v}}), 3),\\
&((\neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg \mathsf{T}_{\mathbf{r}} \wedge \neg \mathsf{T}_{\mathbf{v}}), 2),\\
&((\Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}}), 1),\\
&(\top, 0)).
\end{aligned}
$$

Proposition 24. *Let ϕ be a **QSAT**$_3$-formula in nnf.*

(a) *For any $\mathsf{N} \in \mathcal{N}_{\mathsf{rs}}$, we have that $(s_d, s'_d) \in \mathcal{NE}(\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\widehat{\widehat{\gamma}}_{\mathbf{v}}, \widehat{\widehat{\gamma}}_{\mathbf{r}}))$ for any strategy s_d and $s_{d'}$ with $s_d(q_0) = s'_d(q_0) = \top$.*

(b) *ϕ is satisfiable iff there is an $\mathsf{N} \in \mathcal{N}_{\mathsf{rs}}$ such that for all $s \in \mathcal{NE}(\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\widehat{\widehat{\gamma}}_{\mathbf{v}}, \widehat{\widehat{\gamma}}_{\mathbf{r}}))$ with $out_{\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} (\Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}) \wedge \mathsf{ass} \wedge \mathsf{tick}$.*

(c) *ϕ is satisfiable iff there is an $\mathsf{N} \in \mathcal{N}_{\mathsf{s}}$ such that for all $s \in \mathcal{NE}(\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\widehat{\widehat{\gamma}}_{\mathbf{v}}, \widehat{\widehat{\gamma}}_{\mathbf{r}}))$ with $out_{\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} (\Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}) \wedge \mathsf{ass} \wedge \mathsf{tick}$.*

Proof.

(a) The action profile (s_d, s'_d) yields the path $q_0 q_d^{\omega}$ which satisfies $\bigcirc \mathsf{d}$ if the transition $(\top, \top)$ is not regimented, and the path q_0^{ω} which satisfies $\neg\mathsf{tick}$ if it is regimented. On both paths the payoff for both players is maximal. No player can improve its payoff.

(b) "$\Rightarrow$": Suppose ϕ holds. By Lemma 23 there is an $\mathsf{N} \in \mathcal{N}_{\mathsf{s}}$ such that for all strategies $s_{\mathbf{r}}$ of $\mathbf{r}$ with $s_{\mathbf{r}}(q_0) = \bot$ there is a strategy $s_{\mathbf{v}}$ of $\mathbf{v}$ such that $out_{\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} \mathsf{T}_{\mathbf{r}} \rightarrow (\mathsf{T}_{\mathbf{v}} \wedge \Diamond \mathsf{win}_{\mathbf{v}})$ and for all paths $\lambda \in \Lambda_{\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0)$ we have that $\lambda \models \mathsf{ass} \wedge \mathsf{tick}$. Now suppose that there was a Nash equilibrium $(s_{\mathbf{v}}, s_{\mathbf{r}})$ such that $out_{\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} \mathsf{ass} \wedge \mathsf{tick} \rightarrow (\neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg \bigcirc \mathsf{d})$. As $\mathsf{ass} \wedge \mathsf{tick}$ is true on all paths, this means that $out_{\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} (\neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg \bigcirc \mathsf{d})$ for this Nash equilibrium. We can use the same reasoning as in the proof of Proposition 20(b) to obtain a contradiction. Hence, such a Nash equilibrium cannot exist.

"$\Leftarrow$": Suppose ϕ does not hold. We have to show that for all $\mathsf{N} \in \mathcal{N}_{\mathsf{rs}}$ there is an $s \in \mathcal{NE}(\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\widehat{\widehat{\gamma}}_{\mathbf{v}}, \widehat{\widehat{\gamma}}_{\mathbf{r}}))$ such that ($\star$) $out_{\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} (\mathsf{ass} \wedge \mathsf{tick}) \rightarrow (\neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg \bigcirc \mathsf{d})$.

Firstly, suppose that $\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}$ contains a path $\lambda \in \Lambda_{\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0)$ with $\lambda \models \neg\mathsf{tick}$. This path can be generated by some strategy profile s which satisfies ($\star$), as the antecedent of ($\star$) will be false. As this strategy profile gives maximal utility it is a Nash equilibrium. Thus, from now on we can assume that all paths in the norm updated model satisfy tick. Completely analogous, we can also assume that ass is satisfied in the updated models.

Secondly, consider $\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}$ such that all paths $\lambda \in \Lambda_{\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0)$ satisfy $\mathsf{tick} \wedge \mathsf{ass}$. As ϕ does not hold, by Lemma 23 and our assumption about the updated models we have that for all $\mathsf{N} \in \mathcal{N}_{\mathsf{rs}}$ there is a strategy $s_{\mathbf{r}}$ of $\mathbf{r}$ with $s_{\mathbf{r}}(q_0) = \bot$ such that for all strategies $s_{\mathbf{v}}$ of $\mathbf{v}$, $out_{\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} \mathsf{T}_{\mathbf{r}} \wedge (\mathsf{T}_{\mathbf{v}} \rightarrow \neg\Diamond \mathsf{win}_{\mathbf{v}})$. The rest of the proof follows analogously to Proposition 20(b): let $s'_{\mathbf{v}}$ be any strategy such that $out_{\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, (s'_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} \mathsf{T}_{\mathbf{v}}$ and $s'_{\mathbf{v}}(q_0) = \bot$. By Lemma 23, the verifier cannot change its strategy to ensure $\Diamond \mathsf{win}_{\mathbf{v}}$. Moreover, as $s_{\mathbf{r}}(q_0) = \bot$ the verifier $\mathbf{v}$ cannot deviate to increase its payoff. Analogously, as $s'_{\mathbf{v}}(q_0) = \bot$, the refuter cannot deviate from $s_{\mathbf{r}}$ to increase its payoff. This shows that $(s_{\mathbf{r}}, s'_{\mathbf{v}})$ is indeed a Nash equilibrium. Thus we have that there is an $s \in \mathcal{NE}(\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\widehat{\widehat{\gamma}}_{\mathbf{v}}, \widehat{\widehat{\gamma}}_{\mathbf{r}}))$ with $out_{\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} \neg\Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg \bigcirc \mathsf{d}$.

(c) Follows immediately from (b) and Lemma 23. □

Theorem 25 *(Hardness strong implementation: existence, sanctioning). Let $(\mathcal{I}, q)$ be a pointed NIS and $\mathcal{N} \in \{\mathcal{N}_{\mathsf{rs}}, \mathcal{N}_{\mathsf{s}}\}$. The problem whether $\mathsf{SI}_{\mathcal{NE}}(\mathcal{I}, q, f) \neq \emptyset$ is $\mathbf{\Sigma_3^p}$-hard.*

Proof. First, let $\mathcal{N} = \mathcal{N}_{\mathsf{rs}}$ and ϕ be a **QSAT**$_3$ formula in nnf. We have for $f(\widehat{\widehat{\gamma}}_{\mathbf{v}}, \widehat{\widehat{\gamma}}_{\mathbf{r}}) = (\Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}) \wedge \mathsf{ass} \wedge \mathsf{tick}$:

ϕ satisfiable

iff $\exists \mathsf{N} \in \mathcal{N} (\forall s \in \mathcal{NE}(\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\widehat{\widehat{\gamma}}_{\mathbf{v}}, \widehat{\widehat{\gamma}}_{\mathbf{r}})) : out_{\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} (\Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}) \wedge \mathsf{ass} \wedge \mathsf{tick}$ (Proposition 24 (b))

iff $\exists \mathsf{N} \in \mathcal{N} (\forall s \in \mathcal{NE}(\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\widehat{\widehat{\gamma}}_{\mathbf{v}}, \widehat{\widehat{\gamma}}_{\mathbf{r}})) : out_{\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} (\Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}) \wedge \mathsf{ass} \wedge \mathsf{tick}$

and $\mathcal{NE}(\widehat{\mathfrak{M}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\widehat{\widehat{\gamma}}_{\mathbf{v}}, \widehat{\widehat{\gamma}}_{\mathbf{r}})) \neq \emptyset$) (Proposition 24(a))

iff $\mathsf{SI}_{\mathcal{NE}}(\mathfrak{I}, q_0, f) \neq \emptyset$

The case $\mathcal{N} = \mathcal{N}_s$ follows analogously using Proposition 24(c) in the first step. □

C.2. Regimentation norms

Finally, we consider the case in which $\mathsf{N} \in \mathcal{N}_r$. We have already seen how to ensure that regimenting norms do not regiment specific transitions, by means of the formula tick. However, we can no longer use the previous construction, based on sanctioning norms, to encode a truth assignment. We have to find a way to achieve this with regimenting norms. The idea of this construction consists of two parts:

1. Use regimenting norms to simulate truth assignments of variables x_1 to x_r by removing some of the outgoing transitions of states in $\{q_1, \ldots, q_r\}$.
2. Ensure that no other transition is regimented by using a formula which characterizes the structure of $\widehat{\mathfrak{M}}^{\widehat{\phi}}$.

Definition 28 *(The model $\widehat{\mathfrak{M}}^{\widehat{\phi}}_{\mathbf{r}}$).* Let $\phi \equiv \exists\{x_1, \ldots, x_r\}\forall\{x_{r+1}, \ldots, x_{r+m}\}\exists\{x_{r+m+1}, \ldots, x_n\}\varphi(x_1, \ldots, x_n)$ in nnf be given. We define $\widehat{\mathfrak{M}}^{\widehat{\phi}}_{\mathbf{r}}$ as $\widehat{\mathfrak{M}}^{\widehat{\phi}}$ but each state from $\{q_1, q_{1\bot}, q_{1\top}, \ldots, q_r, q_{r\bot}, q_{r\top}\}$ is additionally labelled by a fresh proposition $\mathsf{r}_{\mathbf{r}}$.

For part 1, we observe that the model contains only the three looping states q_d, $q_\top$, and $q_\bot$. We introduce the formulae

$$\mathsf{tick}_{\mathbf{r}} = \Box(((\mathsf{t} \wedge \neg\mathsf{r}_{\mathbf{r}}) \rightarrow \bigcirc\neg\mathsf{t}) \wedge ((\neg\mathsf{t} \wedge \neg\mathsf{r}_{\mathbf{r}}) \rightarrow \bigcirc\mathsf{t}) \vee \bigcirc(\mathsf{d} \vee \mathsf{win}_{\mathbf{v}} \vee \mathsf{win}_{\mathbf{r}}))$$

$$\mathsf{valid}_{\mathbf{r}} = \Box(((\mathsf{t} \wedge \mathsf{r}_{\mathbf{r}}) \rightarrow \bigcirc\neg\mathsf{t}) \wedge ((\neg\mathsf{t} \wedge \mathsf{r}_{\mathbf{r}}) \rightarrow \bigcirc\mathsf{t})$$

$$\mathsf{tick} = \mathsf{tick}_{\mathbf{r}} \wedge \mathsf{valid}_{\mathbf{r}}$$

As before we use $\mathsf{tick}_{\mathbf{r}}$ to ensure that no regimentation norm is imposed on any state except from possibly $\{q_1, \ldots, q_r\}$. The idea is that some of these transitions in the set may be regimented. A second formula $\mathsf{valid}_{\mathbf{r}}$ is true on a path on which no transition from states in $\{q_1, \ldots, q_r\}$ which are also contained on the path are regimented. Finally, tick represents that no transition on a path is regimented.

Lemma 26. *Let $\mathsf{N} \in \mathcal{N}_{rs}$ be a normative system such that for all paths $\lambda \in \Lambda_{\widehat{\mathfrak{M}}^{\widehat{\phi}}\restriction\mathsf{N}}(q_0)$ it holds that $\lambda \models \mathsf{tick}_{\mathbf{r}}$. Then, N cannot contain a regimentation norm which is applicable in a state of $Q\backslash\{q_d, q_\bot, q_\top, q_1, q_{1\bot}, q_{1\top}, \ldots, q_r, q_{r\bot}, q_{r\top}\}$.*

Lemma 27. *Given a **QSAT**$_3$-formula $\phi \equiv \exists\{x_1, \ldots, x_r\}\forall\{x_{r+1}, \ldots, x_{r+m}\}\exists\{x_{r+m+1}, \ldots, x_n\}\varphi(x_1, \ldots, x_n)$ in nnf we have that ϕ is satisfiable iff there is an $\mathsf{N} \in \mathcal{N}_r$ such that in $\widehat{\mathfrak{M}}^{\widehat{\phi}}_{\mathbf{r}} \restriction \mathsf{N}$ there is a path from q_0 to q_{r+1} (or to q_ϵ if $r = n$), such that for all strategies $s_{\mathbf{r}}$ of $\mathbf{r}$ with $s_{\mathbf{r}}(q_0) = \bot$ there is a strategy $s_{\mathbf{v}}$ of $\mathbf{v}$ such that $out_{\widehat{\mathfrak{M}}^{\widehat{\phi}}_{\mathbf{r}}\restriction\mathsf{N}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} (\mathsf{T}_{\mathbf{r}} \wedge \mathsf{valid}_{\mathbf{r}}) \rightarrow (\mathsf{T}_{\mathbf{v}} \wedge \Diamond\mathsf{win}_{\mathbf{v}})$, and for all paths $\lambda \in \Lambda_{\widehat{\mathfrak{M}}^{\widehat{\phi}}_{\mathbf{r}}\restriction\mathsf{N}}(q_0)$ we have that $\lambda \models \mathsf{tick}_{\mathbf{r}}$.*

Proof. [Sketch] "⇒": Suppose ϕ is satisfiable. Let v be a witnessing valuation of the variables $\{x_1, \ldots, x_r\}$. We consider the normative system N which regiments from q_i, $1 \leq i \leq r$, the transition $(-, \top)$ (resp. $(-, \bot)$) if $v(x_i) = \mathsf{f}$ (resp. $v(x_i) = \mathsf{t}$). Now the truth assignment of $\{x_{r+1}, \ldots, x_{r+m}, \ldots, x_n\}$ induces witnessing strategies following the same reasoning as in Lemma 19. It is also the case that $\mathsf{tick}_{\mathbf{r}}$ is true on all paths starting in q_0.

"⇐": Suppose there is an $\mathsf{N} \in \mathcal{N}_r$ such that for all strategies $s_{\mathbf{r}}$ of $\mathbf{r}$ with $s_{\mathbf{r}}(q_0) = \bot$ there is a strategy $s_{\mathbf{v}}$ of $\mathbf{v}$ such that $out_{\widehat{\mathfrak{M}}^{\widehat{\phi}}_{\mathbf{r}}\restriction\mathsf{N}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} (\mathsf{T}_{\mathbf{r}} \wedge \mathsf{valid}_{\mathbf{r}}) \rightarrow (\mathsf{T}_{\mathbf{v}} \wedge \Diamond\mathsf{win}_{\mathbf{v}})$, and for all paths $\lambda \in \Lambda_{\widehat{\mathfrak{M}}^{\widehat{\phi}}_{\mathbf{r}}\restriction\mathsf{N}}(q_0)$ we have that $\lambda \models \mathsf{tick}_{\mathbf{r}}$ and there is a path from q_0 to q_{r+1} (or to q_ϵ if $r = n$).

By Lemma 26 the transitions of all states in which $\mathsf{r}_{\mathbf{r}}$ does not hold are not affected by the update by the normative system. We can assume wlog that N regiments exactly one outgoing transition for each of the states in $\{q_1, \ldots, q_r\}$ (it cannot regiment both transitions due to the connectivity property). This is so, because otherwise we move the variable x_i, $1 \leq i \leq r$, of which no transition is regimented to the universally quantified part of ϕ. Thus, the normative system N induces a truth assignment v of the variables $\{x_1, \ldots, x_r\}$ as follows: $v(x_i) = \mathsf{t}$ (resp. $v(x_i) = \mathsf{f}$) iff transition $(-, \bot)$ (resp. $(-, \top)$) is regimented in q_i. Now, we can apply a reasoning similar to that of Lemma 19 wrt. $\widehat{\phi}[v]$ and the model $\widehat{\mathfrak{M}}^{\widehat{\phi}[v]}_{\mathbf{r}} \restriction \mathsf{N}$. We get that $\widehat{\phi}[v]$ is satisfiable. The normative systems gives a witnessing truth assignment v of the variables $\{x_1, \ldots, x_r\}$ showing that ϕ is satisfiable. □

In the previous result it was crucial to assume that the normative system does not regiment both of the transitions outgoing of some state $q_1, \ldots, q_r$. In the following we have to ensure that normative systems which do not respect this condition, yield some "bad" Nash equilibrium. Therefore, we define the following preference lists:

$$\gamma_{\mathbf{v}}^* = ((\bigcirc \mathsf{d} \vee \neg \mathsf{tick}, 5),$$
$$((\Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{v}}), 4),$$
$$((\neg \Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg \mathsf{T}_{\mathbf{r}} \wedge \mathsf{T}_{\mathbf{v}}), 3),$$
$$((\neg \Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}} \wedge \mathsf{T}_{\mathbf{v}}), 2),$$
$$((\neg \Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}} \wedge \neg \mathsf{T}_{\mathbf{v}}), 1),$$
$$(\top, 0)),$$
$$\gamma_{\mathbf{r}}^* = ((\bigcirc \mathsf{d} \vee \neg \mathsf{tick}_{\mathbf{r}}, 4),$$
$$((\mathsf{valid}_{\mathbf{r}} \wedge \neg \Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}} \wedge \mathsf{T}_{\mathbf{v}}), 3),$$
$$((\mathsf{valid}_{\mathbf{r}} \wedge \neg \Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg \mathsf{T}_{\mathbf{r}} \wedge \neg \mathsf{T}_{\mathbf{v}}), 2),$$
$$((\mathsf{valid}_{\mathbf{r}} \wedge \Diamond \mathsf{win}_{\mathbf{v}} \wedge \mathsf{T}_{\mathbf{r}}), 1),$$
$$(\top, 0)).$$

Now we are able to show the following result:

Proposition 28. *Let ϕ be a **QSAT**$_3$-formula in nnf and $\mathsf{N} \in \mathcal{N}_{\mathsf{r}}$.*

(a) *We have that $(s_d, s_d') \in \mathcal{NE}(\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\gamma_{\mathbf{v}}^*, \gamma_{\mathbf{r}}^*))$ for any strategy s_d and s_d' with $s_d(q_0) = s_d'(q_0) = \top$.*

(b) *ϕ is satisfiable iff there is an $\mathsf{N} \in \mathcal{N}_{\mathsf{r}}$ such that for all $s \in \mathcal{NE}(\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\gamma_{\mathbf{v}}^*, \gamma_{\mathbf{r}}^*))$ we have that $out_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} (\Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}) \wedge \mathsf{tick}$.*

Proof.

(a) Either we end up in path $q_0 q_d^\omega$ or in q_0^ω. The former satisfies $\mathsf{tick} \wedge \bigcirc \mathsf{d}$, the latter satisfying $\neg \mathsf{tick}$.

(b) "$\Rightarrow$": Suppose ϕ is satisfiable. Then, we can apply Lemma 27. That is, let $\mathsf{N} \in \mathcal{N}_{\mathsf{r}}$ such that in $\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}$ there is a path from q_0 to q_{r+1} (or to q_ϵ if $r = n$), for all strategies $s_{\mathbf{r}}$ of $\mathbf{r}$ with $s_{\mathbf{r}}(q_0) = \bot$ there is a strategy $s_{\mathbf{v}}$ of $\mathbf{v}$ such that $out_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} (\mathsf{T}_{\mathbf{r}} \wedge \mathsf{valid}_{\mathbf{r}}) \rightarrow (\mathsf{T}_{\mathbf{v}} \wedge \Diamond \mathsf{win}_{\mathbf{v}})$ and for all paths $\lambda \in \Lambda_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0)$ we have that $\lambda \models \mathsf{tick}_{\mathbf{r}}$. Now suppose there were a Nash equilibrium $s = (s_{\mathbf{v}}, s_{\mathbf{r}})$ with $out_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} \mathsf{tick} \rightarrow (\neg \Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg \bigcirc \mathsf{d})$.

First, assume that $out_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \not\models^{\mathbf{LTL}} \mathsf{tick}$. Then, we also have $out_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \not\models^{\mathbf{LTL}} \mathsf{tick}_{\mathbf{r}}$ by Lemma 27 and the fact that $\mathsf{valid}_{\mathbf{r}} \wedge \mathsf{tick}_{\mathbf{r}} \rightarrow \mathsf{tick}$ holds on the path. Thus, the refuter would be better of to deviate to a strategy $s_{\mathbf{r}}'$ such that $out_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}}')) \models \mathsf{T}_{\mathbf{r}} \wedge \mathsf{valid}_{\mathbf{r}}$. Second, assume that $out_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} \mathsf{tick}$. Then, also $out_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} \mathsf{valid}_{\mathbf{r}}$. If $s_{\mathbf{v}}(q_0) = \top$ then the refuter can increase its payoff by also deviating to $s_{\mathbf{r}}(q_0) = \top$; so, let us suppose that $s_{\mathbf{v}}(q_0) = \bot$. We can also assume that the verifier plays a consistent strategy, otherwise it would deviate to one. In that case we can also assume that $s_{\mathbf{r}}$ is consistent; otherwise, the refuter can again deviate to a consistent strategy to increase its payoff. Then, however, by Lemma 27 the verifier can deviate to a better strategy that satisfies $\Diamond \mathsf{win}_{\mathbf{v}}$. If $s_{\mathbf{r}}(q_0) = \top$ then we can also assume that $s_{\mathbf{v}}(q_0) = \top$; otherwise, the verifier would deviate to such a strategy. This contradicts the existence of a Nash equilibrium with the above stated properties.

"$\Leftarrow$": Suppose ϕ does not hold. We show that for all $\mathsf{N} \in \mathcal{N}_{\mathsf{r}}$ there is an $s \in \mathcal{NE}(\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\gamma_{\mathbf{v}}^*, \gamma_{\mathbf{r}}^*))$ with $out_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} \mathsf{tick} \rightarrow (\neg \Diamond \mathsf{win}_{\mathbf{v}} \wedge \neg \bigcirc \mathsf{d})$. We suppose, for the sake of contradiction, that this is not the case, i.e. that $(\star)$ for all $s \in \mathcal{NE}(\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\gamma_{\mathbf{v}}^*, \gamma_{\mathbf{r}}^*))$ we have that $out_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} (\Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}) \wedge \mathsf{tick}$.

As ϕ does not hold, we have according to Lemma 27 that for all $\mathsf{N} \in \mathcal{N}_{\mathsf{r}}$ it holds that:

(i) there is no path from q_0 to q_{r+1} (or to q_ϵ if $r = n$) in $\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}$;

(ii) there is a path $\lambda \in \Lambda_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0)$ such that $\lambda \not\models \mathsf{tick}_{\mathbf{r}}$; or

(iii) there is a strategy $s_{\mathbf{r}}$ of $\mathbf{r}$ with $s_{\mathbf{r}}(q_0) = \bot$ such that for all strategies $s_{\mathbf{v}}$ of $\mathbf{v}$ it holds that $(\star)$ $out_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, (s_{\mathbf{v}}, s_{\mathbf{r}})) \models^{\mathbf{LTL}} (\mathsf{T}_{\mathbf{r}} \wedge \mathsf{valid}_{\mathbf{r}}) \wedge (\mathsf{T}_{\mathbf{v}} \rightarrow \neg \Diamond \mathsf{win}_{\mathbf{v}})$.

First, suppose that (i) holds. Consider the strategies $s_{\mathbf{r}}$ and $s_{\mathbf{v}}$ with $s_{\mathbf{r}}(q_0) = s_{\mathbf{v}}(q_0) = \bot$ which end in the looping state in-between q_0 and q_{r+1} (or to q_ϵ if $r = n$) which has to exist by assumption. We have that $out_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} \neg \mathsf{tick} \wedge \neg \mathsf{valid}_{\mathbf{r}}$. Thus, $(s_{\mathbf{v}}, s_{\mathbf{r}})$ is a Nash equilibrium which contradicts $(\star)$.

Secondly, suppose that (ii). As for (i) suppose both players play $s_{\mathbf{r}}$ and $s_{\mathbf{v}}$ with $s_{\mathbf{r}}(q_0) = s_{\mathbf{v}}(q_0) = \bot$ which yield the very path λ with $\lambda \not\models \mathsf{tick}_{\mathbf{r}}$. On this path it holds that $\neg \mathsf{tick}$ and $\neg \mathsf{tick}_{\mathbf{r}}$, thus no player has an incentive to deviate from it.

Thirdly, suppose that (iii) holds. Let $s_{\mathbf{r}}$ be a strategy as defined above, and $s_{\mathbf{v}}$ such that it is consistent and $s_{\mathbf{v}}(q_0) = \bot$. By Lemma 27 the verifier cannot change its strategy to a consistent one which ensures $\Diamond \mathsf{win}_{\mathbf{v}}$. The player would also

not deviate to an inconsistent one. Moreover, no player can deviate to ensure a path with $\bigcirc$d. We can also assume that (i) and (ii) do not hold. Thus, neither the verifier nor the refuter can deviate to a strategy such that the outcome path satisfies $\neg$tick nor $\neg$tick$_{\mathbf{r}}$, respectively. Thus, this strategy profile is a Nash equilibrium, contradicting ($\star$). □

Theorem 29 *(Hardness of strong implementation: existence, regimentation norms). Let* $(\mathfrak{I}, q)$ *be a pointed NIS and* $\mathcal{N} \in \{\mathcal{N}_{\mathrm{r}}\}$. *The problem whether* $\mathsf{SI}_{\mathcal{N}\mathcal{E}}(\mathfrak{I}, q, f) \neq \emptyset$ *is* $\mathbf{\Sigma_3^P}$*-hard.*

Proof. Let ϕ be a **QSAT**$_3$ formula in nnf. We have for $f(\gamma_{\mathbf{v}}^*, \gamma_{\mathbf{r}}^*) = (\Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}) \wedge \mathsf{tick}$:

$$
\begin{aligned}
& \phi \text{ satisfiable} \\
\text{iff } & \exists \mathsf{N} \in \mathcal{N}(\forall s \in \mathcal{NE}(\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\gamma_{\mathbf{v}}^*, \gamma_{\mathbf{r}}^*)) : out_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} (\Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}) \wedge \mathsf{tick} \quad \text{(Proposition 28 (b))} \\
\text{iff } & \exists \mathsf{N} \in \mathcal{N}(\forall s \in \mathcal{NE}(\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\gamma_{\mathbf{v}}^*, \gamma_{\mathbf{r}}^*)) : out_{\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}}(q_0, s) \models^{\mathbf{LTL}} (\Diamond \mathsf{win}_{\mathbf{v}} \vee \bigcirc \mathsf{d}) \wedge \mathsf{tick} \\
& \text{and } \mathcal{NE}(\widehat{\mathfrak{M}}_{\mathbf{r}}^{\widehat{\phi}} \restriction \mathsf{N}, q_0, (\gamma_{\mathbf{v}}^*, \gamma_{\mathbf{r}}^*)) \neq \emptyset) \quad \text{(Proposition 28 (a))} \\
\text{iff } & \mathsf{SI}_{\mathcal{NE}}(\mathfrak{I}, q_0, f) \neq \emptyset \qquad \square
\end{aligned}
$$

References

[1] T. Ågotnes, W. van der Hoek, M. Wooldridge, Normative system games, in: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '07, ACM, New York, NY, USA, 2007, pp. 1–8.
[2] T. Ågotnes, W. van der Hoek, M. Woolridge, Robust normative systems and a logic of norm compliance, Log. J. IGPL 18 (2010) 4–30.
[3] T. Ågotnes, M. Wooldridge, Optimal social laws, in: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1, AAMAS '10, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2010, pp. 667–674, http://dl.acm.org/citation.cfm?id=1838206.1838294.
[4] H. Aldewereld, V. Dignum, Operetta: organization-oriented development environment, in: Languages, Methodologies, and Development Tools for Multi-Agent Systems – Third International Workshop, LADS 2010, Lyon, France, August 30–September 1, 2010, pp. 1–18, revised selected papers.
[5] H. Aldewereld, J. Vázquez-Salceda, F. Dignum, J.C. Meyer, Verifying norm compliancy of protocols, in: O. Boissier, J.A. Padget, V. Dignum, G. Lindemann, E.T. Matson, S. Ossowski, J.S. Sichman, J. Vázquez-Salceda (Eds.), Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems, in: Lecture Notes in Computer Science, vol. 3913, Springer, 2006, pp. 231–245.
[6] N. Alechina, M. Dastani, B. Logan, Programming norm-aware agents, in: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 12, vol. 2, 2012, pp. 1057–1064.
[7] N. Alechina, M. Dastani, B. Logan, Reasoning about normative update, in: Proceedings of the Twenty Third International Joint Conference on Artificial Intelligence, IJCAI 2013, AAAI Press, 2013, pp. 20–26.
[8] N. Alechina, M. Dastani, B. Logan, Norm approximation for imperfect monitors, in: Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2014, 2014.
[9] R. Alur, T.A. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (2002) 672–713.
[10] F. Arbab, Abstract behavior types: a foundation model for components and their composition, in: F. de Boer, M. Bonsangue, S. Graf, W.-P. de Roever (Eds.), Formal Methods for Components and Objects, in: LNCS, vol. 2852, Springer-Verlag, 2003, pp. 33–70.
[11] L. Astefanoaei, M. Dastani, J.-J.C. Meyer, F. Boer, On the semantics and verification of normative multi-agent systems, Int. J. Univers. Comput. Sci. 15 (13) (2009) 2629–2652.
[12] J. Baumfalk, B. Poot, B. Testerink, M. Dastani, A sumo extension for norm based traffic control systems, in: Proceedings of the SUMO2015 Intermodal Simulation for Intermodal Transport, DLR, Berlin, Adlershof, 2015, pp. 63–83.
[13] P. Blackburn, J. Bos, K. Striegnitz, Learn Prolog Now!, Texts in Computing, vol. 7, College Publications, 2006.
[14] G. Boella, L. van der Torre, Regulative and constitutive norms in normative multiagent systems, in: Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning, KR'04, 2004, pp. 255–266.
[15] G. Boella, L.W.N. van der Torre, Enforceable social laws, in: The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, AMAAS 2005, 2005, pp. 682–689.
[16] O. Boissier, R. Bordini, J. Hübner, A. Ricci, A. Santi, Multi-agent oriented programming with jacamo, Sci. Comput. Program 78 (6) (2011) 747–761.
[17] F. Brazier, C. Jonker, J. Treur, Compositional design and reuse of a generic agent model, Appl. Artif. Intell. J. 14 (2000) 491–538.
[18] N. Bulling, M. Dastani, Normative mechanism design (extended abstract), in: Proceedings of the 10th International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2011, ACM Press, Taipei, Taiwan, May 2011, pp. 1187–1188.
[19] N. Bulling, M. Dastani, Verification and implementation of normative behaviours in multi-agent systems, in: Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence, IJCAI, Barcelona, Spain, July 2011, pp. 103–108.
[20] N. Bulling, M. Dastani, M. Knobbout, Monitoring norm violations in multi-agent systems, in: Twelfth International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'13, 2013, pp. 491–498.
[21] N. Bulling, J. Dix, Modelling and verifying coalitions using argumentation and ATL, Intel. Artif. 14 (46) (March 2010) 45–73.
[22] N. Bulling, W. Jamroga, Verifying agents with memory is harder than it seemed, AI Commun. 23 (4) (December 2010) 389–403.
[23] N. Bulling, W. Jamroga, J. Dix, Reasoning about temporal properties of rational play, Ann. Math. Artif. Intell. 53 (1–4) (2009) 51–114.
[24] H.L. Cardoso, E. Oliveira, Electronic institutions for b2b: dynamic normative environments, Artif. Intell. Law 16 (1) (2008) 107–128.
[25] M. Comuzzi, I. Vanderfeesten, T. Wang, Optimized cross-organizational business process monitoring: design and enactment, Inf. Sci. 244 (2013) 107–118.
[26] N. Criado, E. Argente, V. Botti, Open issues for normative multi-agent systems, AI Commun. 24 (3) (2011) 233–264.
[27] M. Dastani, J.-J.C. Meyer, D. Grossi, A logic for normative multi-agent programs, J. Log. Comput. 23 (2) (2013) 335–354.
[28] M. Dastani, N. Tinnemeier, J.-C. Meyer, A programming language for normative multi-agent systems, in: V. Dignum (Ed.), Multi-Agent Systems: Semantics and Dynamics of Organizational Models, Information Science Reference, 2009.
[29] F. Dignum, Autonomous agents with norms, Artif. Intell. Law 7 (1) (1999) 69–79, http://dx.doi.org/10.1023/A%3A1008315530323.
[30] U. Endriss, S. Kraus, J. Lang, M. Wooldridge, Designing incentives for boolean games, in: AAMAS, 2011, pp. 79–86.

[31] M. Esteva, D. de la Cruz, C. Sierra, ISLANDER: an electronic institutions editor, in: Proceedings of the First International Joint Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2002, Bologna, Italy, 2002, pp. 1045–1052.
[32] M. Esteva, J. Rodríguez-Aguilar, B. Rosell, J. Arcos, AMELI: an agent-based middleware for electronic institutions, in: Proceedings of AAMAS 2004, New York, US, July 2004, pp. 236–243.
[33] M. Esteva, J. Rodriguez-Aguilar, C. Sierra, W. Vasconcelos, Verifying norm consistency in electronic institutions, in: V. Dignum, D. Corkill, C. Jonker, F. Dignum (Eds.), Proceedings of the AAAI-04 Workshop on Agent Organizations: Theory and Practice, AAAI, AAAI Press, San Jose, July 2004, Technical Report WS-04-02.
[34] D. Fitoussi, M. Tennenholtz, Choosing social laws for multi-agent systems: minimality and simplicity, Artif. Intell. 119 (1) (2000) 61–101.
[35] N. Fornara, M. Colombetti, Specifying and enforcing norms in artificial institutions, in: Proc. of DALT'08, 2009.
[36] A. Garcia-Camino, P. Noriega, J.A. Rodriguez-Aguilar, Implementing norms in electronic institutions, in: Proceedings of the Fourth International Joint Conference on Autonomous Agents and MultiAgent Systems, AAMAS'05, New York, NY, USA, 2005, pp. 667–673.
[37] G. Gottlob, G. Greco, F. Scarcello, Pure Nash equilibria: hard and easy games, J. Artif. Intell. Res. (2003) 215–230.
[38] H. Guo, Automotive Informatics and Communicative Systems: Principles in Vehicular Networks and Data Exchange, Information Science Reference – Imprint of IGI Publishing, Hershey, PA, 2009.
[39] J. Hintikka, Logic, Language Games and Information, Clarendon Press, Oxford, 1973.
[40] R. Horowitz, P. Varaiya, Control design of an automated highway system, in: Special Issue on Hybrid Systems, Proc. IEEE 88 (7) (2000) 913–925.
[41] J. Hübner, O. Boissier, R. Kitio, A. Ricci, Instrumenting multi-agent organisations with organisational artifacts and agents: giving the organisational power back to the agents, Int. J. Auton. Agents Multi-Agent Syst. 20 (2010) 369–400.
[42] J. Hübner, J. Sichman, O. Boissier, $\mathcal{S}$–$\mathcal{M}$OISE$^+$: a middleware for developing organised multi-agent systems, in: Proceedings of the International Workshop on Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems, in: LNCS, vol. 3913, Springer, 2006, pp. 64–78.
[43] J. Hübner, J. Sichman, O. Boissier, Developing organised multiagent systems using the $\mathcal{M}$OISE$^+$ model: programming issues at the system and agent levels, Int. J. Agent-Oriented Softw. Eng. 1 (3/4) (2007) 370–395.
[44] J.F. Hübner, O. Boissier, R.H. Bordini, From organisation specification to normative programming in multi-agent organisations, in: J. Dix, J. Leite, G. Governatori, W. Jamroga (Eds.), Proceedings of 11th International Workshop on Computational Logic in Multi-Agent Systems, CLIMA XI, Lisbon, Portugal, August 16–17, 2010, in: Lecture Notes in Computer Science, vol. 6245, Springer, 2010, pp. 117–134.
[45] A.J.I. Jones, M. Sergot, On the characterization of law and computer systems, in: J.-J.C. Meyer, R. Wieringa (Eds.), Deontic Logic in Computer Science: Normative System Specification, John Wiley & Sons, 1993, pp. 275–307.
[46] D. Krajzewicz, J. Erdmann, M. Behrisch, L. Bieker, Recent development and applications of sumo-simulation of urban mobility, Int. J. Adv. Syst. Meas. 5 (3&4) (2012).
[47] K. Mahbub, G. Spanoudakis, A framework for requirements monitoring of service based systems, in: Proceedings of the 2nd International Conference on Service Oriented Computing, ICSOC '04, ACM, New York, NY, USA, 2004, pp. 84–93.
[48] F.R. Meneguzzi, M. Luck, Norm-based behaviour modification in BDI agents, in: C. Sierra, C. Castelfranchi, K.S. Decker, J.S. Sichman (Eds.), 8th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2009, IFAAMAS, 2009, pp. 177–184.
[49] A. Metzger, P. Leitner, D. Ivanovic, E. Schmieders, R. Franklin, M. Carro, S. Dustdar, K. Pohl, Comparing and combining predictive business process monitoring techniques, IEEE Trans. Syst. Man Cybern. 45 (2) (2014) 276–290.
[50] J. Meyer, R. Wieringa, Deontic Logic in Computer Science: Normative System Specification, Wiley Professional Computing, J. Wiley, 1993.
[51] N.H. Minsky, V. Ungureanu, Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems, ACM Trans. Softw. Eng. Methodol. 9 (3) (2000).
[52] J. Morales, M. Lopez-Sanchez, J.A. Rodriguez-Aguilar, M. Wooldridge, W. Vasconcelos, Minimality and simplicity in the on-line automated synthesis of normative systems, in: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems, AAMAS '14, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2014, pp. 109–116.
[53] J. Morales, M. López-Sánchez, J.A. Rodríguez-Aguilar, M. Wooldridge, W.W. Vasconcelos, Automated synthesis of normative systems, in: International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA, May 6–10, 2013, pp. 483–490.
[54] M. Osborne, A. Rubinstein, A Course in Game Theory, MIT Press, 1994.
[55] C. Papadimitriou, Computational Complexity, Addison Wesley, Reading, 1994.
[56] A. Pnueli, The temporal logic of programs, in: Proceedings of Foundations of Computer Science, FOCS, 1977, pp. 46–57.
[57] A. Ricci, M. Viroli, A. Omicini, Give agents their artifacts: the A&A approach for engineering working environments in MAS, in: E.H. Durfee, M. Yokoo, M.N. Huhns, O. Shehory (Eds.), 6th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2007, IFAAMAS, 2007.
[58] Y. Shoham, K. Leyton-Brown, Multiagent Systems – Algorithmic, Game-Theoretic, and Logical Foundations, Cambridge University Press, 2009.
[59] Y. Shoham, M. Tennenholtz, On the synthesis of useful social laws for artificial agent societies, in: Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI-92, San Diego, CA, 1992.
[60] Y. Shoham, M. Tennenholtz, On social laws for artificial agent societies: off-line design, Artif. Intell. 73 (1–2) (1995) 231–252.
[61] V.T. Silva, From the specification to the implementation of norms: an automatic approach to generate rules from norms to govern the behavior of agents, Int. J. Auton. Agents Multiagent Syst. 17 (1) (2008) 113–155.
[62] M.P. Singh, M. Arrott, T. Balke, A.K. Chopra, R. Christiaanse, S. Cranefield, F. Dignum, D. Eynard, E. Farcas, N. Fornara, F. Gandon, G. Governatori, H.K. Dam, J. Hulstijn, I. Krueger, H.-P. Lam, M. Meisinger, P. Noriega, B.T.R. Savarimuthu, K. Tadanki, H. Verhagen, S. Villata, The uses of norms, in: G. Andrighetto, G. Governatori, P. Noriega, L.W.N. van der Torre (Eds.), Normative Multi-Agent Systems, in: Dagstuhl Follow-Ups, vol. 4, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2013, pp. 191–229.
[63] I. Sommerville, D. Cliff, R. Calinescu, J. Keen, T. Kelly, M. Kwiatkowska, J. Mcdermid, R. Paige, Large-scale complex it systems, Commun. ACM 55 (7) (2012) 71–77, http://doi.acm.org/10.1145/2209249.2209268.
[64] N. Tinnemeier, M. Dastani, J.-J.C. Meyer, L. van der Torre, Programming normative artifacts with declarative obligations and prohibitions, in: Proceedings of IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, IEEE Computer Society, 2009, pp. 145–152.
[65] W. van der Hoek, M. Roberts, M. Wooldridge, Social laws in alternating time: effectiveness, feasibility, and synthesis, Synthese 156 (1) (2007) 1–19.
[66] M.B. van Riemsdijk, K.V. Hindriks, C.M. Jonker, Programming organization-aware agents: a research agenda, in: Proceedings of the Tenth International Workshop on Engineering Societies in the Agents' World, ESAW'09, in: LNAI, vol. 5881, Springer, 2009, pp. 98–112.
[67] K.W. Wagner, Bounded query classes, SIAM J. Comput. 19 (5) (1990) 833–846.
[68] M. Wooldridge, An Introduction to Multiagent Systems, 2nd edition, Wiley, Chichester, UK, 2009.
[69] M. Wooldridge, U. Endriss, S. Kraus, J. Lang, Incentive engineering for boolean games, Artif. Intell. 195 (2013) 418–439, http://www.sciencedirect.com/science/article/pii/S0004370212001518.
[70] F. Zambonelli, N. Jennings, M. Wooldridge, Organizational abstractions in the analysis and design of multi-agent systems, in: First International Workshop on Agent-Oriented Software Engineering at ICSE, 2000.