Gavel: A sanctioning enforcement framework[⋆]

Igor Conrado Alves de Lima[1,3], Luis Gustavo Nardin[2], and Jaime Simão Sichman[3]

[1] Inst. de Matemática e Estatística, Universidade de São Paulo, São Paulo SP, Brazil
[2] Dept. of Informatics, Brandenburg University of Technology, Cottbus, Germany
[3] LTI, Escola Politécnica, Universidade de São Paulo, São Paulo SP, Brazil
`igorcadelima@usp.br, nardin@b-tu.de, jaime.sichman@poli.usp.br`

Abstract. Sanctioning is one of the most adopted enforcement mechanisms in the governance of multiagent systems. Current enforcement frameworks, however, restrict agents to reason about and make sanctioning decisions. We developed the Gavel framework, an adaptive sanctioning enforcement framework that enables agents to decide for the most appropriate sanction to apply depending on various decision factors. The potential benefits and use of the framework are shown using a Public Goods Game in which agents are endowed with different strategies combining material and reputational sanctions.

Keywords: Enforcement System · Normative Multiagent Systems · Software Engineering.

1 Introduction

Norm enforcement is one of the central puzzles in social order and social control theories. It refers to the process in which an entity monitors and encourages others to comply with norms. Sanctioning is one of the most adopted and largely recognised norm enforcement mechanisms used to promote appropriate behavioural standards, in particular norm compliance [26]. Norm enforcement, specially sanctioning, has been addressed in a broad range of perspectives and disciplines, such as philosophy [3], law [16], economics [4], sociology [17] and social psychology [9]. These disciplines recognise that different sanction types (e.g., emotional, informational, reputational, and material [28]) are used by individuals and institutions to enforce and promote norms compliance.

In Normative Multiagent Systems (NMASs), norm enforcement enables reaction to norms violation (i.e., punishment) or compliance (i.e., reward) henceforth identified as *sanction.* The degree to which a norm is enforced plays a crucial role in NMAS dynamics and conveys a great deal of norm-relevant information that affects other normative processes.

There are two traditional approaches to norm enforcement in NMAS[4]: *regimentation* and *regulation.* The former assumes that agents can be controlled

[⋆] Igor Conrado Alves de Lima was fully supported by CNPq, Brazil, grant number 131120/2016-6.

[4] See [2] for an extended taxonomy of norm enforcement mechanisms.

and non-compliant actions prevented. The latter allows violations, yet sanctions may be applied to the violator whenever a violation is detected.

Both approaches can be arranged in a centralised or distributed mode. The regimentation approach operates mostly in a centralised mode through normative institution frameworks, such as Electronic Institutions [25, 13] and Organisation Models [12, 14]. These frameworks provide a reference normative system to which agents have to abide and infrastructure entities enforce the compliance of agents' actions and interactions with the norms. The regulation approach, conversely, operates mostly in a distributed mode and equires that agents' architectures, such as BOID [7], NoA [18], and EMIL-A [1], are endowed with mechanisms that enable agents to enforce norms.

Cardoso and Oliveira [8] proposed a centralised norm enforcement mechanism for contractual commitments. Their mechanism pre-define sanctions that are applied by enforcer agents without taking into account any individual or contextual information. Centeno et al. [10] extended this approach to adapt sanctions based on contextual information. Modgil et al [23] proposed a general distributed architecture for norm-governed systems that relies on distributed infrastructural agents to monitor and apply pre-defined sanctions. In line with López and Luck [20], Criado et al. [11] relaxed some of the constraints imposed on the infrastructural enforcer agents allowing them to punish or reward due to, respectively, norms' violation or compliance. In this mechanism, each norm is associated with specific punishment or reward sanctions, thus limiting the agents' decision autonomy. To overcome this limitation, Villatoro et al. [30] proposed a technique that allows enforcers to adapt the strength of the sanction based on the number of defectors. Mahmoud et al. [21] proposed the use of the violation characteristics to adapt the magnitude or frequency of the sanction. Moreover, Mahmoud et al. [22] introduced the use of reputation as a means for enforcers to adapt the strength of the sanctions.

Although Pasquier et al. [27] identified the importance and need to have different sanction types and endow agents with sanction reasoning and decision capabilities, Nardin et al. [24] showed that the available norm enforcement frameworks lack full support to four main requirements to render these features possible:

- **R1** *Support for multiple categories of sanctions* (e.g., legal sanctions, ostracism, reputation spreading);
- **R2** *Potential association of multiple sanctions with a norm violation or compliance* (e.g., provide a set of sanction options instead of pre-establishing a fixed set to a norm);
- **R3** *Reasoning about the most adequate sanction to apply depending on different factors* (e.g., one might consider the sanctionee's history to determine an appropriate sanction to apply, if any); and
- **R4** *Adaption of the sanction content depending on context* (e.g., a norm violation of high magnitude might incur a more severe negative sanction).

In Nardin et al. [24], the authors propose a conceptual sanctioning process model that could possibly overcome these drawbacks. However, they have not designed or implemented an adaptive sanctioning enforcement framework based on this conceptual model. This is precisely what we present in this paper: the development of the Gavel framework, based on this previous conceptual model. Moreover, we show the potential benefits and use of this framework through a Public Goods Game (PGG) [19], in which agents are endowed with different strategies combining material and reputational sanctions.

2 Gavel Framework

Gavel is an adaptive sanctioning enforcement framework based on the conceptual sanctioning process model presented by Nardin et al. [24]. It enables agents to decide for the most appropriate sanctions to apply, depending on their current context assessed by a set of sanctioning decision factors.

The conceptual sanctioning process model specifies the features and components of our sanctioning enforcement framework. Both norm violation and compliance are considered in the process, respecting the general notion of sanction as a negative or positive reaction to normative behaviours. The entire sanctioning process is realised by agents endowed with special capabilities (i.e., Detector, Evaluator, Executor, Controller, and Legislator) supported by specialised data repositories (*De Jure* and *De Facto*). Next, we define the components of our norm enforcement framework.

2.1 NMAS

Definition 1. (NMAS) *A NMAS is a system composed of a set of autonomous and heterogeneous agents situated in a shared environment, whose actions and interactions are ruled by norms and sanctions. A NMAS, either open or closed, is defined as*

$$\text{NMAS} = \langle \mathcal{E}nv, \mathcal{A}g, \mathcal{R}, \mathcal{A}c, \mathcal{N}, \mathcal{S}, \mathcal{L} \rangle,$$

where

- $\mathcal{E}nv$ *is the environment that may assume any of a finite set of discrete states;*
- $\mathcal{A}g = \{ag_i : i \leq |\mathcal{A}g|\}$ *is the set of agents that can act on the environment or interact among themselves;*
- $\mathcal{R} = \{r_i : i \leq |\mathcal{R}|\}$ *is the set of roles that agents can play;*
- $\mathcal{A}c = \{\alpha_i : i \leq |\mathcal{A}c|\}$ *is the set of actions that agents can perform;*
- $\mathcal{N} = \{n_i : i \leq |\mathcal{N}|\}$ *is the set of norms prescribing the agents' behaviours;*
- $\mathcal{S} = \{s_i : i \leq |\mathcal{S}|\}$ *is the set of sanctions prescribing possible reactions to norm violation or compliance;*
- $\mathcal{L} = \mathcal{N} \times \mathcal{S}$ *is the set of links between norms and sanctions.*

2.2 Norms, Sanctions and Links

Definition 2. (Norm) *A norm* $\mathrm{n_i} \in \mathcal{N}$ *is a guide of conduct prescribing how agents ought to behave in a given situation. A norm is defined as*

$$\mathrm{n_i} = \langle \text{status, activation, issuer, target, deactivation, deadline, content} \rangle,$$

where

- status $\in$ {enabled, disabled} *indicates whether* $\mathrm{n_i}$ *is in force;*
- activation *is the set of contextual conditions that renders the norm applicable;*
- issuer $\in \mathcal{A}g$ *identifies the entity that originally issued the norm;*
- target $\in \mathcal{A}g$ *identifies the agent to which the norm is addressed;*
- deactivation *is the set of contextual conditions that renders the norm no longer applicable once active;*
- deadline *is the set of contextual and temporal conditions which determine the deadline to comply with the norm;*
- content *is the criteria prescribing the agents' behaviours.*

Definition 3. (Norm Instance) *A norm instance* $\mathrm{n_i}'$ *is the result of applying a ground substitution to a norm* $\mathrm{n_i}$*. A norm instance is defined as*

$$\mathrm{n_i}' = \langle \text{status}', \text{activation}', \text{issuer}', \text{target}', \text{deactivation}', \text{deadline}', \text{content}' \rangle,$$

where each term of $\mathrm{n_i}'$ *unifies with its corresponding in* $\mathrm{n_i}$*.*

Definition 4. (Sanction) *A sanction* $\mathrm{s_i} \in \mathcal{S}$ *is a reaction to a norm compliance or violation. A sanction is defined as*

$$\mathrm{s_i} = \langle \text{status}, \text{activation}, \text{category}, \text{content} \rangle,$$

where

- status $\in$ {enabled, disabled} *indicates whether* $\mathrm{s_i}$ *is in force;*
- activation *is the set of contextual conditions that renders the sanction applicable;*
- category *is the sanction classification according to the sanction typology detailed in [24], defined as*

  $$\text{category} = \langle \text{purpose}, \text{issuer}, \text{locus}, \text{mode}, \text{polarity}, \text{discernability} \rangle,$$

  where
  - purpose $\in$ {Punishment, Reward, Enablement, Guidance, Incapacitation}*,*
  - issuer $\in$ {Formal, Informal}*,*
  - locus $\in$ {Self-Directed, Other-Directed}*,*
  - mode $\in$ {Direct, Indirect}*,*
  - polarity $\in$ {Positive, Negative}*,*
  - discernability $\in$ {Noticeable, Unnoticeable}*;*
- content *is the specification of the set of actions representing the sanction.*

Definition 5. (Sanction Instance) *A sanction instance* $\mathrm{s_i}'$ *is the result of applying a ground substitution to a sanction* $\mathrm{s_i}$*. A sanction instance is defined as*

$$\mathrm{s_i}' = \langle \mathrm{status}', \mathrm{activation}', \mathrm{category}', \mathrm{content}' \rangle,$$

where each term of $\mathrm{s_i}'$ *unifies with its corresponding in* $\mathrm{s_i}$.

Definition 6. (Link) *A link* $\mathrm{l_i} \in \mathcal{L}$ *is an association between a norm and a subset of sanctions. A link is defined as*

$$\mathrm{l_i} = \langle \mathrm{n_i}, \mathcal{SL}_{\mathrm{n_i}} \rangle,$$

where

- $\mathrm{n_i} \in \mathcal{N}$ *is the norm being linked;*
- $\mathcal{SL}_{n_i} = \{\mathrm{sl_j} \mid \mathrm{sl_j} = \langle \mathrm{status}, \mathrm{s_j} \rangle\}$ *is the set of sanction links to* $\mathrm{n_i}$*, where*
  - $\mathrm{status} \in \{\mathrm{enabled, disabled}\}$ *indicates whether* $\mathrm{sl_j}$ *is in force and*
  - $\mathrm{s_j} \in \mathcal{S}$ *is the sanction being linked.*

An enabled link states that an agent may consider a sanction $\mathrm{s_j}$ *as a possible reaction to the compliance or violation of the norm* $\mathrm{n_i}$.

2.3 Repositories

We define two types of data repositories: *De Jure* and *De Facto*.

Definition 7. (De Jure) De Jure *($\mathcal{DJ}$) is a repository which stores specifications of norms and sanctions and their associations. It is defined as*

$$\mathcal{DJ} = \langle \mathcal{N}^{\mathcal{DJ}}, \mathcal{S}^{\mathcal{DJ}}, \mathcal{L}^{\mathcal{DJ}} \rangle,$$

where

- $\mathcal{N}^{\mathcal{DJ}} \subseteq \mathcal{N}$ *is the set of all norms stored in* $\mathcal{DJ}$*;*
- $\mathcal{S}^{\mathcal{DJ}} \subseteq \mathcal{S}$ *is the set of all sanctions stored in* $\mathcal{DJ}$*;*
- $\mathcal{L}^{\mathcal{DJ}} \subseteq \mathcal{L}$ *is the set of all links between norms and sanctions stored in* $\mathcal{DJ}$.

Definition 8. (De Facto) De Facto *($\mathcal{DF}$) is a repository of historical information about sanction decisions, applications, and outcomes. It is defined as*

$$\mathcal{DF} = \langle \mathcal{SD}^{\mathcal{DF}}, \mathcal{SA}^{\mathcal{DF}}, \mathcal{SO}^{\mathcal{DF}} \rangle,$$

where

- $\mathcal{SD}^{\mathcal{DF}}$ *(*Sanction Decision Set*) represents the set of sanction decisions made by* Evaluators *and stored in* $\mathcal{DF}$*. Each sanction decision* $\mathrm{sd_i} \in \mathcal{SD}^{\mathcal{DF}}$ *is defined as*

  $$\mathrm{sd_i} = \langle \mathrm{time_d}, \mathrm{detector}, \mathrm{evaluator}, \mathrm{target}, \mathrm{n_j}', \mathrm{s_k}', \mathrm{cause} \rangle,$$

  where
  - $\mathrm{time_d}$ *indicates the global time at which the sanction was decided;*

- detector $\in \mathcal{A}g$ *identifies the agent that reported the norm compliance or violation;*
- evaluator $\in \mathcal{A}g$ *identifies the agent that decided the sanction;*
- target $\in \mathcal{A}g$ *identifies the agent to which the sanction is directed;*
- $\mathrm{n_j}'$ *is the norm instance which was evaluated by the* evaluator*;*
- $\mathrm{s_k}'$ *is the sanction decided by* evaluator *for* target *in response to* $\mathrm{n_j}'$*;*
- cause $\in$ {compliance, violation} *indicates what led* evaluator *to decide for the sanction* $\mathrm{s_k}'$*.*

– $\mathcal{SA}^{\mathcal{DF}}$ *(*Sanction Application Set*) represents the set of sanction applications executed by* Executors*. Each sanction application* $\mathrm{sa_i} \in \mathcal{SA}^{\mathcal{DF}}$ *is defined as*

$$\mathrm{sa_i} = \langle \mathrm{time_a}, \mathrm{decision_j}, \mathrm{executor} \rangle,$$

where

- $\mathrm{time_a}$ *indicates the global time at which the sanction was applied;*
- $\mathrm{decision_j} \in \mathcal{SD}^{\mathcal{DF}}$ *is the sanction decision to which* $\mathrm{sa_i}$ *is related;*
- executor $\in \mathcal{A}g$ *identifies the agent that applied the sanction.*

– $\mathcal{SO}^{\mathcal{DF}}$ *(*Sanction Outcome Set*) represents the set of sanction outcomes observed by a* Controller*. Each sanction outcome* $\mathrm{so_i} \in \mathcal{SO}^{\mathcal{DF}}$ *is defined as*

$$\mathrm{so_i} = \langle \mathrm{time_o}, \mathrm{application_j}, \mathrm{controller}, \mathrm{efficacy} \rangle,$$

where

- $\mathrm{time_o}$ *indicates the global time at which the efficacy of the sanction was assessed;*
- $\mathrm{application_j} \in \mathcal{SA}^{\mathcal{DF}}$ *is the observed sanction application;*
- controller $\in \mathcal{A}g$ *identifies the agent that observed the outcome;*
- efficacy *indicates how effective the sanction was in promoting norm compliance. It can use discrete (e.g.,* effective *and* ineffective*) or continuous (e.g.,* $[-1, 1]$*) values.*

2.4 Capabilities

The Gavel framework defines five capabilities: Detector, Evaluator, Executor, Controller, and Legislator. Agents having these capabilities perform tasks in different stages of the sanctioning process.

Definition 9. (Detector) *The* Detector *perceives the environment and detects a norm violation or compliance. It* watches *for normative events,* creates *norm instances, and* reports *compliances and violations to an* Evaluator*. The* watch *function is defined as*

$$\mathrm{watch} : \mathrm{e} \times \mathcal{KB} \times \mathcal{N}_{\mathrm{enabled}} \rightarrow \mathcal{N}', \qquad (1)$$

where

– e *is the event to be analysed;*
– $\mathcal{KB}$ *is the* Detector*'s knowledge base;*

- $\mathcal{N}_{\text{enabled}} = \{\text{n}_\text{i} \mid \text{n}_\text{i} \in \mathcal{N} \wedge \text{n}_\text{i}.\text{status} = \text{enabled}\}$ *is the set of enabled norms known by the agent;*
- $\mathcal{N}' = \{\text{n}_\text{i}' \mid \text{n}_\text{i} \in \mathcal{N}_{\text{enabled}}$ *and* $\text{e} \wedge \mathcal{KB} \models \text{n}_\text{i}'.\text{activation}\}$ *is the set of norm instances whose activation condition holds given* e *and* $\mathcal{KB}$*.*

Each norm instance obtained from watch *is assessed as complied, violated, or deactivated. If complied or violated, then the* Detector *reports such fact to an* Evaluator*.*

Definition 10. (Evaluator) *The* Evaluator *receives from the* Detector *the report of a violation or compliance of a norm instance* n_i'*. It then obtains from the* De Jure *repository all the applicable sanctions associated with* n_i' *by enabled links to decide for the sanctions it judges appropriate to apply, if any. This task is performed by the* evaluate *function, which is defined as*

$$\text{evaluate} : \text{n}_\text{i}' \times \mathcal{KB} \times \mathcal{SL}_{\text{n}_\text{i}',\text{enabled}} \rightarrow \mathcal{SD}^{\mathcal{DF}}_{\text{n}_\text{i}'}, \tag{2}$$

where

- n_i' *is the norm instance to be evaluated;*
- $\mathcal{KB}$ *is the knowledge base from which the agent extracted contextual factors to be considered in the evaluation;*
- $\mathcal{SL}_{\text{n}_\text{i}',\text{enabled}}$ *is the set of enabled sanction links associated with* n_i'*;*
- $\mathcal{SD}^{\mathcal{DF}}_{\text{n}_\text{i}'}$ *is the set of sanction decisions for* n_i'*.*

Definition 11. (Executor) *The* Executor *agent* ag_i *receives from the* Evaluator *a sanction decision* $\text{sd}_\text{j} \in \mathcal{SD}$ *and decides whether or not to* execute *it. The decision for not executing a sanction could result either from lack of resources to operate or personal interests. In a real-world setting, for example,* Evaluators *and* Executors *would be comparable to judges and police officers, respectively. The* execute *function maps a sanction decision to actions in the environment:*

$$\text{execute} : \text{sd}_\text{j} \rightarrow \mathcal{A}c. \tag{3}$$

If the actions defined in sd_j *are successfully executed, then the* Executor *records the sanction application* $\text{sa}_\text{k} = \langle \text{time}_\text{a}, \text{sd}_\text{j}, \text{ag}_\text{i} \rangle$ *in the* $\mathcal{SA}^{\mathcal{DF}}$*.*

Definition 12. (Controller) *The* Controller ag_i *monitors the outcomes of a sanction application* sa_k *to determine its efficacy and records its judgement as a sanction outcome* $\text{so}_\text{j} = \langle \text{time}_\text{o}, \text{sa}_\text{k}, \text{ag}_\text{i}, \text{efficacy} \rangle$ *in* $\mathcal{SO}^{\mathcal{DF}}$*.*

Definition 13. (Legislator) *The* Legislator *creates, removes, and updates norms, sanctions, and their associations in* De Jure *based on the assessment of the* De Jure *and* De Facto *repositories along with its knowledge base.*

$$\text{legislate} : \mathcal{DJ} \times \mathcal{DF} \times \mathcal{KB} \rightarrow \mathcal{DJ} \tag{4}$$

3 Implementation

The Gavel framework[5] has been implemented in Java and is mainly divided into three packages (see Figure 1):

gavel.api provides interfaces for all the elements of the model;
gavel.base provides abstract classes which can be used as basis for customisation of some elements of the model;
gavel.impl contains generic concrete implementations of the model elements (e.g., norms, sanctions, norm-sanction links, and data repositories) according to contracts prescribed by interfaces defined in *gavel.api* and provides utility classes with factory, parsing, and other supporting methods.

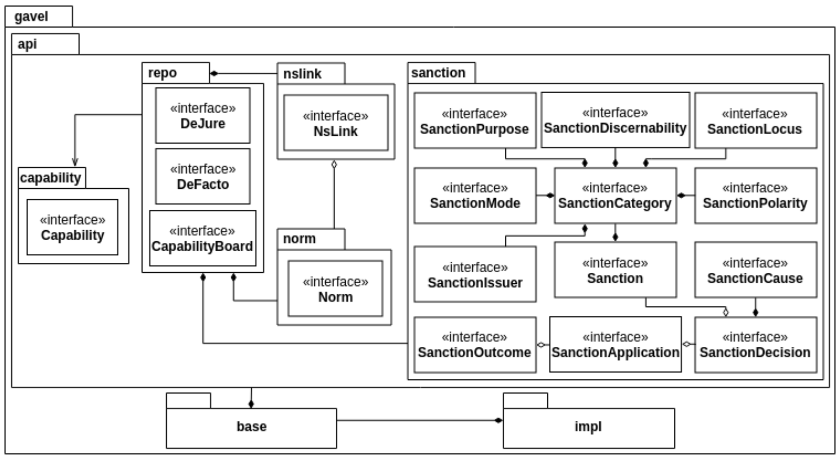

gavel
api
repo
«interface» DeJure
«interface» DeFacto
«interface» CapabilityBoard
capability
«interface» Capability
nslink
«interface» NsLink
norm
«interface» Norm
sanction
«interface» SanctionPurpose
«interface» SanctionDiscernability
«interface» SanctionLocus
«interface» SanctionMode
«interface» SanctionCategory
«interface» SanctionPolarity
«interface» SanctionIssuer
«interface» Sanction
«interface» SanctionCause
«interface» SanctionOutcome
«interface» SanctionApplication
«interface» SanctionDecision
base
impl

Fig. 1: *Gavel*'s architecture.

The framework includes generic *in-memory* data storage implementations for three data repositories:

`DeJure` stores and provides operations to manage norms, sanctions, and norm-sanction links. Initial norms, sanctions and norm-sanction links may be loaded into `DeJure` by means of a *regulative specification* file. At runtime, these elements can be created, retrieved, updated, or deleted;
`DeFacto` stores sanction decisions, applications, and outcomes at runtime;
`CapabilityBoard` stores capability assignment rules and the capabilities possessed by agents. If an agent has a certain capability and the repository is informed, then such information will be available for the entire system. Also, initial capability assignment rules may be loaded into `CapabilityBoard` via a *capability assignment specification* file.

It is often desirable to specify norms, sanctions, norm-sanction links, and capability assignment rules before the system starts running. Thus, system designers can provide two XML (eXtensible Markup Language) specification files:

[5] Source code available at https://github.com/gavelproject/gavel/.

– *Regulative specification* — defines norms, sanctions, and norm-sanction links specifications that will be initially stored in `DeJure`;
– *Capability assignment specification* — defines rules specifying which agents will be allowed to possess which of the capabilities defined in the model.

Notice that Gavel does not provide execution plans for each capability as these are dependent on the multiagent system platform.

In addition to the standard Java implementation, we have designed and implemented *Gavel for JaCaMo*[6], a reusable framework which integrates Gavel with JaCaMo [5]. The benefit of such integration is twofold: *(i)* Gavel repositories are provided as CArtAgO [29] artefacts that may be used by agents; and *(ii)* since Jason supports meta-programming [6], agents may acquire plans at runtime from `CapabilityBoard` to learn how to perform tasks inherent to any of the sanctioning capabilities (Detector, Evaluator, Executor, Controller, Legislator). We have used this implementation in our case study, presented next.

4 Case Study

We have used the *Gavel for JaCaMo* framework to implement a version of the Public Goods Game (PGG) partially inspired by [15].

4.1 Public Goods Game Model

Broadly used in experimental economics, agents in the PGG have private tokens and secretly choose whether to contribute to a public pool. The tokens in this pool are multiplied by a benefit factor and evenly divided among players.

In our PGG model, agents are endowed with a number of tokens and play the game for a number of rounds (see Algorithm 1) or until they are in deficit of tokens. At each round, agents are randomly grouped (line 2) and they decide whether to free-ride or contribute a fixed amount to the public pool (line 3). The sum of the contributions in each group is multiplied by a benefit factor and evenly divided among the group agents regardless of their contribution (line 4). Next, the agents' decisions are disclosed to all other agents in their group (line 5) and agents decide whether or not to apply sanctions to other agents in their group (line 6). Once sanctions are applied (line 7), agents with less than zero tokens are eliminated from the game (line 10–12).

Agents can have one of four types of contribution strategies:

– *Cooperator* (C) who always contributes to the pubic pool and does not sanction other agents;
– *Free-Rider* (FR) who never contributes to the public pool and does not sanction other agents;
– *Nice* (N) who always contributes to the public pool and may apply sanctions if the percentage of detected free-riders in its group exceeds a threshold;

[6] Source code available at https://github.com/gavelproject/gavel-jacamo/.

Algorithm 1 Public Goods Game main cycle

```
 1: Initialise agents
 2: for number of rounds do
 3:     Random group formation
 4:     Agents make their contribution decision
 5:     Gather and distribution of contributions in each group
 6:     Disclose contribution decisions in the group
 7:     Agents make their sanction decisions
 8:     Apply sanctions
 9:     for each agent do
10:         if Agent's tokens < 0 then
11:             Agent is culled from the game
12:         end if
13:     end for
14: end for
```

– *Mean* (M) who decides to free-ride and may apply sanctions if the percentage of detected free-riders in its group exceeds a threshold; otherwise, it contributes and does not sanction.

An agent ag_i identifies an agent ag_j as free-rider if the reputation that ag_i has about ag_j is below a certain threshold. Agents keep an individual record of all other agents in the game. Each agent ag_i calculates the reputation of ag_j by taking the weighted average of its direct experience and the reputation received from others about ag_j, which is defined as

$$R_{ij} = W \times \Delta E + (1 - W) \times \Delta I, \tag{5}$$

where $R_{ij} \in [0, 1]$ is the reputation the agent ag_i has about the agent ag_j, where 0 means the worst and 1 means the best reputation. ΔE is the proportion of good personal experiences ag_i had with ag_j, ΔI is the average reputation received about ag_j by ag_i, and W is the weight given to the personal experiences.

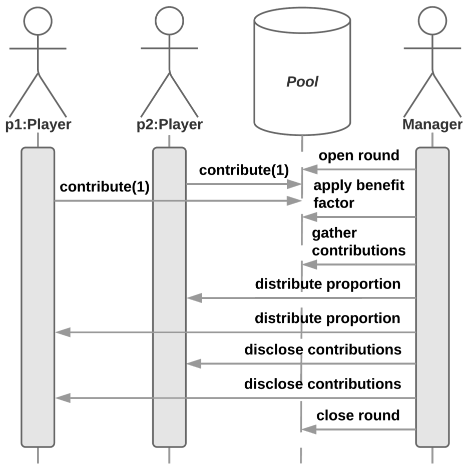
Nice and Mean agents also use features of the Gavel model to guide their sanction choice towards free-riders. The sanction strategies available are:

– *Random* (R): Agents decide randomly between gossiping about or punishing free-riders;
– *Threshold* (T): Agents decide whether to gossip or punish by comparing the reputation of the free-rider with a randomly picked number from a uniform distribution between 0 and 1. If the free-rider's reputation is less than the random number, the agent punishes the free-rider, otherwise it gossips.

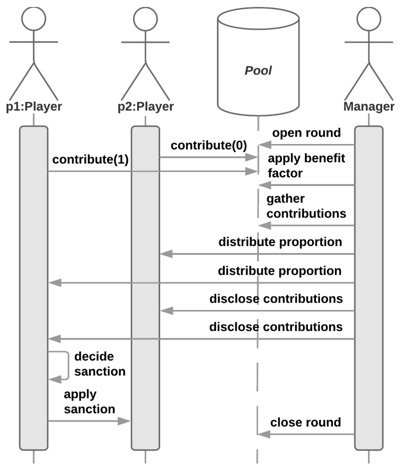
There are some constraints to apply either type of sanction. Each agent has a limit on the number of reputation transmissions in each round. If this limit is reached, reputation transmission is not possible. The punishment inflicted on free-riders has a cost to the agent inflicting it, thus an agent can only sanction if it can afford. This cost, called *enforcement cost*, could be seen as the effort required to apply a sanction. It is worth noticing that agents do not lie in this model, thus only truthful information is transmitted and only defectors are punished.

4.2 Agents Interaction

Figure 2a depicts a fully norm-compliant round of the game. Once the manager opens the round, players start contributing to the pool. When all contributions are made, the manager applies the benefit factor, gathers the resulting amount, distributes each agent's portion, discloses contributions, and closes the round.



(a) Round with all players complying with the norm.



(b) Round with one player violating the norm.

Fig. 2: Sequence diagrams of a norm-compliant and a non-norm-compliant round.

Figure 2b illustrates a round in which a sanctioning occurs. After the manager discloses the contributions, the player *p1* notices that *p2* did not contribute. Then, player *p1* decides for a sanction and applies it to the player *p2*.

4.3 Implementation

For the simulation of our PGG[7], we have implemented two types of agents: game *manager* and *player*. The manager is responsible for 1) creating rounds; 2) defining groups; 3) gathering contributions; 4) multiplying the total contribution by a benefit factor; 5) dividing the result evenly among players; and 6) disclosing the contribution of each player. Conversely, players are limited to 1) contributing to the pool; and 2) sanctioning other players based on the content of `DeJure`. The pools to which players cooperate are controlled by the manager and implemented as domain artefacts using CArtAgO.

All players are endowed with the Detector, Evaluator, Executor, and Controller capabilities, but for the sake of simplicity, no Legislator was included.

Only one norm regulates the players' behaviours in our PGG. As shown below, this norm, identified as `positive_contribution`, states that every player

[7] Source code available at https://github.com/gavelproject/pgg/.

is obliged to contribute with 1 token to every pool it participates. If an agent does not comply with the norm before the pool finishes, then the norm is violated.

```
norm(
  id(positive_contribution),
  status(enabled),
  activation(pool_member(Player)),
  issuer(manager),
  target(Player),
  deactivation(false),
  deadline(pool_status("FINISHED")),
  content(obligation(contribution(Player,1)))
)
```

The following two links state that the `positive_contribution` norm is linked to two sanctions, `punishment` and `gossip`:

```
ns_link(
  nid(positive_contribution),
  sanction_links(sanction_link(status(enabled),sid(punishment)),
                 sanction_link(status(enabled),sid(gossip))
  )
)
```

The `punishment` sanction is only applicable if the player evaluating the violation is not the target and can afford the sanction. As shown below, this negative informal sanction counts as applied when the Executor directly punishes the target inflicting a pre-established cost.

```
sanction(
  id(punishment),
  status(enabled),
  activation(not .my_name(Target) & cost_to_punish(Cost) & tokens(Tokens) & Cost <= Tokens ),
  category(purpose(punishment), issuer(informal), locus(other_directed), mode(direct),
           polarity(negative), discernability(noticeable)),
  content(punish(Target))
)
```

Conversely, the `gossip` sanction is only applicable if the player evaluating the violation is not the target and has not reached the limit of reputation transmissions in that round, there is a player in another group, and the target is considered a free-rider. As shown below, this is a negative informal sanction that counts as applied when the Executor transmits reputation about the target to another agent from another group.

```
sanction(
  id(gossip),
  status(enabled),
  activation(not .my_name(Target) & not transmissions_credit(0) &
             not players_in_other_groups([]) &
             reputation(Target,Reputation) &
             min_reputation_cooperator(MinRepCoop) &
             Reputation < MinRepCoop),
  category(purpose(punishment), issuer(informal), locus(other_directed), mode(indirect),
           polarity(negative), discernability(unnoticeable)),
  content(gossip(Target,Reputation))
)
```

4.4 Evaluation Scenarios

We ran 2 evaluation scenarios to analyse how Gavel enables agents to reason about sanctions. These scenarios vary by just one feature, the type of sanctioning strategy (i.e., Random or Threshold) employed by all agents. For each scenario the agents population was formed by 400 agents, 100 of each contribution strategy (i.e., Cooperators, Free-riders, Nice, and Mean). The contribution to the public pool was set to 1 token, and the benefit factor was set to 3. Each agent was endowed with an initial amount of 50 tokens to be used to contribute to the public pool or to sanction others. Nice and Mean agents consider sanctioning other agents if they detect more than 20% of free-riders in their group. An agent is considered a free-rider if its reputation is below a threshold set to 0.6. A punishment involves a cost to the punisher (i.e., enforcement cost) and a cost to the punished agent (i.e., punishment cost). A gossip does not have a cost, although its use is limited by a maximum number of transmissions per round.

We ran the model for 100 rounds in each scenario, repeating 10 times with different random seeds for each combination of parameter values from Table 1.

Table 1: Simulation parameters

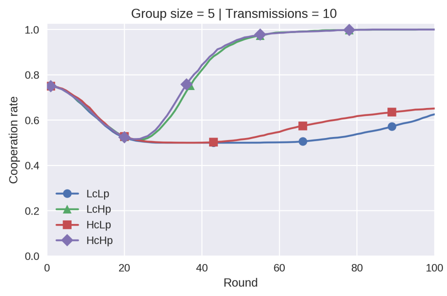
Parameter	Values	
Enforcement cost	0.2	1
Punishment cost	2	5
Group size	5	
Number of transmissions	10	

In our scenarios we aimed at showing the potential benefits and use of Gavel to implement a simple (i.e., Random) and a more elaborated (i.e., Threshold) sanctioning strategies. We evaluated these scenarios by checking the average proportion of cooperation per group measured as the total number of agents contributing to the pool divided by the total number of agents per round. Four combinations of enforcement and punishment costs were identified: LcLp (low cost, low punishment), LcHp (low cost, high punishment), HcLp (high cost, low punishment), and HcHp (high cost, high punishment).

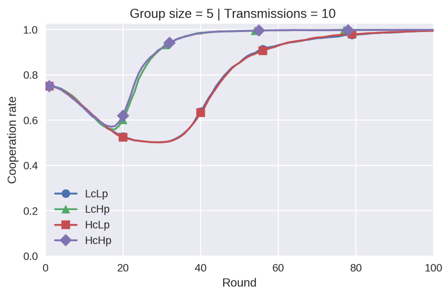
We started by evaluating the scenario in which agents employ the Random strategy. As shown in Figure 3a, the agents were not able to achieve more than 70% of cooperation when the punishment cost was low. Starting approximately from the 70^{th} round, however, both LcHp and HcHp allowed cooperation to reach 100% as a result from the death of free-riders.

Figure 3b shows that higher levels of cooperation were achieved when agents employed the Threshold strategy. Due to an informed heuristic used in this approach, agents were able to achieve 100% of cooperation for every combination of parameters. For the LcHp and HcHp combinations, 100% could even be achieved earlier when compared to Random.

Our results show that the sanction reasoning capability provided by Gavel allows agents to adapt to their current context improving the effectiveness of



(a) Random sanction strategy



(b) Threshold sanction strategy

Fig. 3: Cooperation rates when agents employ different sanction strategies.

their actions and, specifically to PGG, it helps to improve the cooperation rate.

5 Conclusions and Future Work

In this paper, we have designed and implemented an adaptive sanctioning enforcement framework, called Gavel, based on the conceptual model proposed by Nardin et al. [24]. We implemented *Gavel for JaCaMo*, an integration of Gavel with JaCaMo, and used it to implement a version of the *Public Goods Game* (PGG) inspired by [15], in which agents can decide which type of sanction to apply at each stage of the game. Our results show that the Threshold sanction strategy, a simple sanctioning decision heuristic that uses reputation, improves the cooperation rate in the game compared to the Random sanction strategy, a sanctioning strategy that does not make any informed decision for sanctioning.

The main advantages for using Gavel are its flexibility and adaptability. Gavel can be treated as a component which can be connected to or implemented within any agent. By using it, agents are free to choose the sanctions and intensity they deem the best to sanction a violator or complier agent. They may also update the legislation, or *De Jure*, to obtain higher levels of norm compliance. As these decisions are all dependent on the current context and historical facts, Gavel can, therefore, assure high level of flexibility and adaptability for norm enforcement in NMAS.

Conversely, Gavel's main disadvantages are limited control and predictability of final results. These are actually direct consequences of the flexibility it provides. As the sanctioning mechanism depends on the system's history and evolution, this influences how agents will learn and apply sanctions.

Our next main step is to further explore the adaptability Gavel provides. We intend to use reinforcement learning to allow Evaluators making better sanction decisions and Legislators updating De Jure based on De Facto. Furthermore, we plan to conduct experiments using different parameter values (e.g., group sizes) and dissociating cooperation from sanctioning strategy.

References

1. Andrighetto, G., Villatoro, D.: Beyond the carrot and stick approach to enforcement: An agent-based model. In: Kokinov, B., Karmiloff-Smith, A., Nersessian, N.J. (eds.) European Perspectives on Cognitive Science. pp. 1–6. New Bulgarian University Press, Sofia (2011)
2. Balke, T.: A taxonomy for ensuring institutional compliance in utility computing. In: Boella, G., Noriega, P., Pigozzi, G., Verhagen, H. (eds.) Normative Multi-Agent Systems. pp. 1–17. No. 09121 in Dagstuhl Seminar Proceedings, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, Dagstuhl (2009)
3. Beccaria, M., Ingraham, E.D.: An essay on crimes and punishments. Philip H. Nicklin, Philadelphia (No. 175, Chesnut St.) (1819)
4. Becker, G.S.: Crime and punishment: An economic approach. Journal of Political Economy **76**(2), 169–217 (1968)
5. Boissier, O., Hübner, J.F., Ricci, A.: The JaCaMo framework. In: Social coordination frameworks for social technical systems, pp. 125–151. Springer, Cham (2016)
6. Bordini, R.H., Hübner, J.F., Wooldridge, M.J.: Programming multi-agent systems in AgentSpeak using Jason. Wiley Series in Agent Technology, John Wiley & Sons, Chichester (2007)
7. Broersen, J., Dastani, M., Hulstijn, J., Huang, Z., van der Torre, L.: The BOID architecture: Conflicts between beliefs, obligations, intentions and desires. In: Proceedings of the 5th International Conference on Autonomous Agents. pp. 9–16. ACM Press, New York, NY (2001)
8. Cardoso, H.L., Oliveira, E.: Adaptive deterrence sanctions in a normative framework. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology. pp. 36–43. IEEE Computer Society, Washington, D.C. (2009)
9. Carlsmith, K.M., Darley, J.M., Robinson, P.H.: Why do we punish?: Deterrence and just deserts as motives for punishment. Journal of Personality and Social Psychology **83**(2), 284–299 (2002)
10. Centeno, R., Billhardt, H., Hermoso, R.: An adaptive sanctioning mechanism for open multi-agent systems regulated by norms. In: Proceedings of the IEEE 23rd International Conference on Tools with Artificial Intelligence. pp. 523–530. IEEE Computer Society, Washington, D.C. (2011)
11. Criado, N., Argente, E., Noriega, P., Botti, V.: MaNEA: A distributed architecture for enforcing norms in open MAS. Engineering Applications of Artificial Intelligence **26**(1), 76–95 (2013)
12. Dignum, V.: A model for organizational interaction: Based on agents, founded in logic. Ph.D. thesis, Utrecht University (2004)
13. Esteva, M., Rodríguez-Aguilar, J.A., Arcos, J.L., Sierra, C., Garcia, P.: Institutionalizing open multi-agent systems. In: Proceedings of the 4th International Conference on Multi-Agent Systems. pp. 381–382. IEEE Computer Society, Boston, MA (2000)
14. Gâteau, B., Boissier, O., Khadraoui, D., Dubois, E.: MOISE^{Inst}: An organizational model for specifying rights and duties of autonomous agents. In: Gleizes, M.P., Kaminka, G.A., Nowé, A., Ossowski, S., Tuyls, K., Verbeeck, K. (eds.) Proceedings of the 3rd European Workshop on Multi-Agent Systems. pp. 484–485. Koninklijke Vlaamse Academie van Belie voor Wetenschappen en Kunsten, Brussels (2005)
15. Giardini, F., Paolucci, M., Villatoro, D., Conte, R.: Punishment and gossip: Sustaining cooperation in a public goods game. In: Kamiński, B., Koloch, G. (eds.)

Advances in Social Simulation. Advances in Intelligent Systems and Computing, vol. 229, pp. 107–118. Springer, Berlin (2014)

16. Hart, H.L.A.: Punishment and responsibility. Clarendon Press, Oxford, UK (1968)
17. Horne, C.: The rewards of punishment: A relational theory of norm enforcement. Stanford University Press, Palo Alto, CA, US (2009)
18. Kollingbaum, M.J., Norman, T.J.: NoA - a normative agent architecture. In: Gottlob, G., Walsh, T. (eds.) Proceedings of the 18th International Conference on Artificial Intelligence. pp. 1465–1466. Morgan Kaufmann, San Francisco, CA (2003)
19. Ledyard, J.: Public goods: A survey of experimental research. In: Kagel, J.H., Roth, A.E. (eds.) The Handbook of Experimental Economics, pp. 111–194. Princeton University Press, Princeton, NJ (1995)
20. López, F.L.y., Luck, M.: Modelling norms for autonomous agents. In: Chávez, E., Favela, J., Mejía, M., Oliart, A. (eds.) Proceedings of the 4th Mexican International Conference on Computer Science. pp. 238–245. IEEE Computer Society, Washington, D.C. (2003)
21. Mahmoud, S., Griffiths, N., Keppens, J., Luck, M.: Efficient norm emergence through experiential dynamic punishment. In: Raedt, L., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F. (eds.) Proceedings of the 20th European Conference on Artificial Intelligence - Including Prestigious Applications of Artificial Intelligence System Demonstrations Track. Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 576–581. IOS Press, Monpellier (2012)
22. Mahmoud, S., Villatoro, D., Keppens, J., Luck, M.: Optimised reputation-based adaptive punishment for limited observability. In: Proceedings of the 6th IEEE International Conference on Self-Adaptive and Self-Organizing Systems. pp. 129–138. IEEE Computer Society, Washington, DC (2012)
23. Modgil, S., Faci, N., Meneguzzi, F., Oren, N., Miles, S., Luck, M.: A framework for monitoring agent-based normative systems. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems. pp. 153–160. IFAAMAS, Richland, SC (2009)
24. Nardin, L.G., Balke, T., Ajmeri, N., Kalia, A.A., Sichman, J.S., Singh, M.P.: Classifying sanctions and designing a conceptual sanctioning process model for socio-technical systems. The Knowledge Engineering Review **31**(2), 142–166 (2016)
25. Noriega, P.: Agent mediated auctions: The fishmarket metaphor. Ph.D. thesis, Universitat Autònoma de Barcelona (1997)
26. Ostrom, E., Walker, J., Gardner, R.: Covenants with and without a sword: Self-governance is possible. American Political Science Review **86**, 404–417 (1992)
27. Pasquier, P., Flores, R.A., Chaib-draa, B.: Modelling flexible social commitments and their enforcement. In: Gleizes, M.P., Omicini, A., Zambonelli, F. (eds.) Proceedings of the 5th international conference on Engineering Societies in the Agents World (ESAW'04). Lecture Notes in Computer Science, vol. 3451, pp. 139–151. Springer, Berlin (2005)
28. Posner, R.A., Rasmusen, E.B.: Creating and enforcing norms, with special reference to sanctions. International Review of Law and Economics **19**(3), 369–382 (1999)
29. Ricci, A., Piunti, M., Viroli, M., Omicini, A.: Environment programming in CArtAgO. In: Multi-Agent Programming: Languages, Tools and Applications, pp. 259–288. Springer, Boston, MA (2009)
30. Villatoro, D., Andrighetto, G., Sabater-Mir, J., Conte, R.: Dynamic sanctioning for robust and cost-efficient norm compliance. In: Proceedings of the 22th International Joint Conference on Artificial Intelligence. pp. 414–419. AAAI Press, Menlo Park, CA (2011)