Norm identification in multi-agent societies

Bastin Tony Roy Savarimuthu, Stephen Cranefield, Maryam A. Purvis, Martin K. Purvis

Department of Information Science, University of Otago, P O Box 56, Dunedin, New Zealand

Abstract

In normative multi-agent systems, the question of "how an agent identifies a norm in an agent society" has not received much attention. This paper aims at addressing this question. To this end, this paper proposes an architecture for norm identification for an agent. The architecture is based on observation of interactions between agents. This architecture enables an autonomous agent to identify the norms in a society using the Candidate Norm Inference (CNI) algorithm. The CNI algorithm uses association rule mining approach to identify sequences of events as candidate norms. When a norm changes, the agent using our architecture will be able to modify the norm and also remove a norm if it does not hold in its society. Using simulations we demonstrate how an agent makes use of the norm identification framework.

Key words: norms, agents, architecture, identification, simulation, societies

1. Introduction

Most works on norms in normative multi-agent systems have concentrated on how norms regulate behaviour (e.g. [1, 2]). These works assume that the agent somehow knows (*a priori*) what the norms of a society are. For example, an agent may have obtained the norm from a leader [3] or through an institution that prescribes what the norms of the society should be [4, 5, 6].

Only a few researchers have dealt with how an agent may infer what the norms of a newly joined society are [7, 8]. Recognizing the norms of a society is beneficial to an agent. This process enables the agent to know what is permissible within a society and what is not. As the agent joins and leaves different agent societies, this capability is essential for the agent to modify its expectations of behaviour, depending upon the society of which it is a part. As the environment changes, the capability of recognizing a new norm helps an agent to derive new ways of achieving its intended goals. Such a norm identification mechanism can be useful for software agents that need to adapt to changing environment. In open agent systems, instead of possessing predetermined notions of what are the norms, agents can infer and identify norms through observing patterns of

interactions and their consequences. This work aims to answer the question of how agents infer norms in a multi-agent society. To that end, we propose an internal agent architecture for norm identification. The architecture is based on observation of interactions between agents. It enables an autonomous agent to identify the norms in a society using the Candidate Norm Inference (CNI) algorithm presented here. When a norm changes, the agent using our architecture will be able to modify their internal representation of norms and also remove a norm if it does not hold in its society. Using simulations we demonstrate how an agent makes use of the norm identification framework.

The paper is organized as follows. Section 2 provides a background on norms in human societies and how the concept of norms is investigated in the field of normative multi-agent systems (NorMAS). Section 3 provides an overview of the norm identification framework. Section 4 describes the components of the framework. In sections 5 and 6 experimental results on norm identification are presented. Section 7 provides a discussion on the work that has been achieved and the issues that need to be addressed in the future. Concluding remarks are presented in section 8.

2. Background and related work

2.1. Background on norms in human societies

Due to multi-disciplinary interest in norms, several definitions for norms exist. Ullman-Margalit [9] describes a social norm as a prescribed guide for conduct or action which is generally complied with by the members of the society. She states that norms are the resultant of complex patterns of behaviour of a large number of people over a protracted period of time. Elster notes the following about social norms [10]. *"For norms to be social, they must be shared by other people and partly sustained by their approval and disapproval. They are sustained by the feelings of embarrassment,anxiety, guilt and shame that a person suffers at the prospect of violating them. A person obeying a norm may also be propelled by positive emotions like anger and indignation ... social norms have a grip on the mind that is due to the strong emotions they can trigger".*

Based on the definitions provided by various researchers, we note that the notion of a social norm is generally made up of the following three aspects:

- Normative expectation of a behavioural regularity: There is a general agreement within the society that a behaviour is *expected* on the part of an agent (or actor) by others in a society, in a given circumstance.
- Norm enforcement mechanism: When an agent does not follow a norm, it could be subjected to a sanction. The sanction could include monetary or physical punishment in the real world which can trigger emotions (embarrassment, guilt, etc.) or direct loss of utility. Other kind of sanctions could include agents not being willing to interact with an agent that violated the norm or the decrease of its reputation score.

- Norm spreading mechanism: Examples of norm spreading mechanisms include the notion of advice from powerful leaders, imitation and learning on the part of an agent.

Many social scientists have studied why norms are adhered to. Some of the reasons for norm adherence include:

- fear of authority or power
- rational appeal of the norms
- emotions such as shame, guilt and embarrassment that arise because of non-adherence.
- willingness to follow the crowd

2.2. Normative multi-agent systems

The definition of normative multi-agent systems as described by the researchers involved in the NorMAS 2007 workshop is as follows [11]. *A normative multi-agent system is a multi-agent system organized by means of mechanisms to represent, communicate, distribute, detect, create, modify and enforce norms, and mechanisms to deliberate about norms and detect norm violation and fulfillment.*

Researchers in multi-agent systems have studied how the concept of norms can be applied to artificial agents. Norms are of interest to multi-agent system (MAS) researchers as they help in sustaining social order and increase the predictability of behaviour in the society. Researchers have shown that norms improve cooperation and collaboration [12, 13]. Epstein has shown that norms reduce the amount of computation required to make a decision [14]. However, software agents may tend to deviate from norms due to their autonomy. So, the study of norms has become important to MAS researchers as they can build robust multi-agent systems using the concept of norms and also experiment on how norms may evolve and adapt in response to environmental changes.

Research in normative multi-agent systems can be categorized into two branches. The first branch focuses on normative system architectures, norm representations, norm adherence and the associated punitive or incentive measures. Lopez *et al.*[15] have designed an architecture for normative BDI agents. Boella *et al.*[16] have proposed a distributed architecture for normative agents. Some researchers have used deontic logic to define and represent norms [5, 16]. Several researchers have worked on mechanisms for norm compliance and enforcement [17, 4, 18]. A recent development is the research on emotion-based mechanisms for norm enforcement [19, 20]. Conte and Castelfranchi [21] have worked on an integrated view of norms. Their work tries to bridge the gap between the prescriptive view of norms and the emergence of conventions from mere regularities using the cognitive abilities of an agent.

The second branch of research is related to emergence of norms. Several researchers have worked on both prescriptive (top-down) and emergent (bottom-up) approaches to norms. In a top-down approach an authoritative leader or

a normative advisor prescribes what the norm of the society should be [22]. In the bottom-up approach, the agents come up with a norm through learning mechanisms [12, 23]. Researchers have used sanctioning mechanisms [18] and reputation mechanisms [24] for enforcing norms. An overview of different mechanisms used by researchers for the research on norms is provided by Savarimuthu and Cranefield [25].

The work reported in this paper falls under the bottom-up approach in the study of norms. Many researchers in this approach have experimented with game-theoretical models for norm emergence [18, 12]. Agents using these models learn to choose a strategy that maximizes utility. The agents in these works do not possess the notion of "normative expectation". Several researchers have proposed architectures for normative systems. For a comparison of these architectures refer to Neumann's article [26]. These architectures assume that norms exist in the society and the focus is on how the norms can be regulated in an institutional setting. Very few have investigated how an agent comes to know the norms of the society. Our objective in this work is to propose an architecture where agents can identify what the norms of the society are.

We note that our work parallels the work that is being carried out by the researchers involved in the EMIL project [7]. Researchers involved in the EMIL project [7] are working on a cognitive architecture for norm emergence. They aim to deliver a simulation-based theory of norm innovation, where norm innovation is defined as the two-way dynamics of an inter-agent process and an intra-agent process. The inter-agent process results in the emergence of norms where the micro interactions produce macro behaviour (norms). The intra-agent process refers to what goes inside an agent's mind so that it can recognize what the norms of the society are. This approach uses cognitive agents that examine interactions between agents and are able to recognize what the norms could be. The agents in this model need not necessarily be utility maximizing like the ones in the learning models. The agents in the model will have the ability to filter external requests that affect normative decisions and will also be able to communicate about norms with other agents. Agents just employing learning algorithms lack these capabilities.

The work reported here differs from this work in three ways. Firstly, in our architecture we have chosen "reaction" or "signalling" (positive and negative) to be a top-level construct for identifying potential norms when the norm of a society is being shaped. We note that a sanction not only may imply a monetary punishment, it could also be an action that could invoke emotions (such as an agent yelling at another might invoke shame or embarrassment on another agent), which can help in norm spreading. Agents can recognize such actions based on their previous experience. Secondly, based on association rule mining [27], we propose an algorithm for norm inference, called the Candidate Norm Inference (CNI) algorithm, which can be adapted by an autonomous agent for flexible norm identification. Thirdly, we identify two different sets of norms in an agent's mind: candidate norms and identified norms.

3. Overview of the framework for norm identification

In this section we provide an overview of the norm identification framework (called the norm engine) that we propose for an agent to infer norms in the agent society in which it is situated. Figure 1 shows the architectural diagram of the norm identification framework. An agent's norm engine is made up of several components. The circles represent information storage components. The rounded boxes represent information processing components, and the diamonds represent decision making components, and the lines represent the flow of information between the components.

An agent employing this architecture follows a six-step process.

Step 1: An agent actively perceives the events in the environment in which it is situated.

Step 2: When an agent perceives an event, it stores the event in its belief base. The events observed by an observer are of two types: regular events and signalling events. In the context of enjoying a public park, a regular event is an event, such as an agent moving to another location in a park or sitting on a bench. Special events are signalling events that agents understand to be either encouraging or discouraging certain behaviour. For example when an agent litters in the park, another agent can discourage the littering action by shouting at the litterer. The signal in this context is the shouting action. We assume that an agent has the ability to recognize signalling events based on its previous experience.

Step 3: When a special event occurs, the agent stores the special event in the special events base. It should be noted that all events are stored in an agent's belief base but only special events are stored in the special events base.

Step 4: If the perceived event is a special event an agent checks if there exists a norm in its *personal norm* (*p-norm*) base or the *group norm* (*g-norm*) base. An agent may possess some p-norms[1] based on its past experience or preference. A *p-norm* may vary across agents, since a society may be made up agents with different backgrounds and experiences. A *g-norm* is a norm which an agent infers, based on its personnel interactions as well as the interactions it observes in the society. An agent infers g-norms using the norm inference component.

When a special event occurs an agent may decide to invoke its norm inference component to identify whether a previously unknown norm may have resulted in the occurrence of the special event. In the context of a park scenario, when an agent observes an agent yelling at another agent, it invokes the norm inference component to find out what events that had happened in the past may have triggered the occurrence of the special event. In other words an agent is interested to find out whether the special event can be explained by the existence of a norm in the society. The invocation of the norm inference

[1] A *p-norm* is the personal value of an agent. For example an agent may consider that littering is an action that should be prohibited in a society. This personal value may not be shared by the agents in a society.

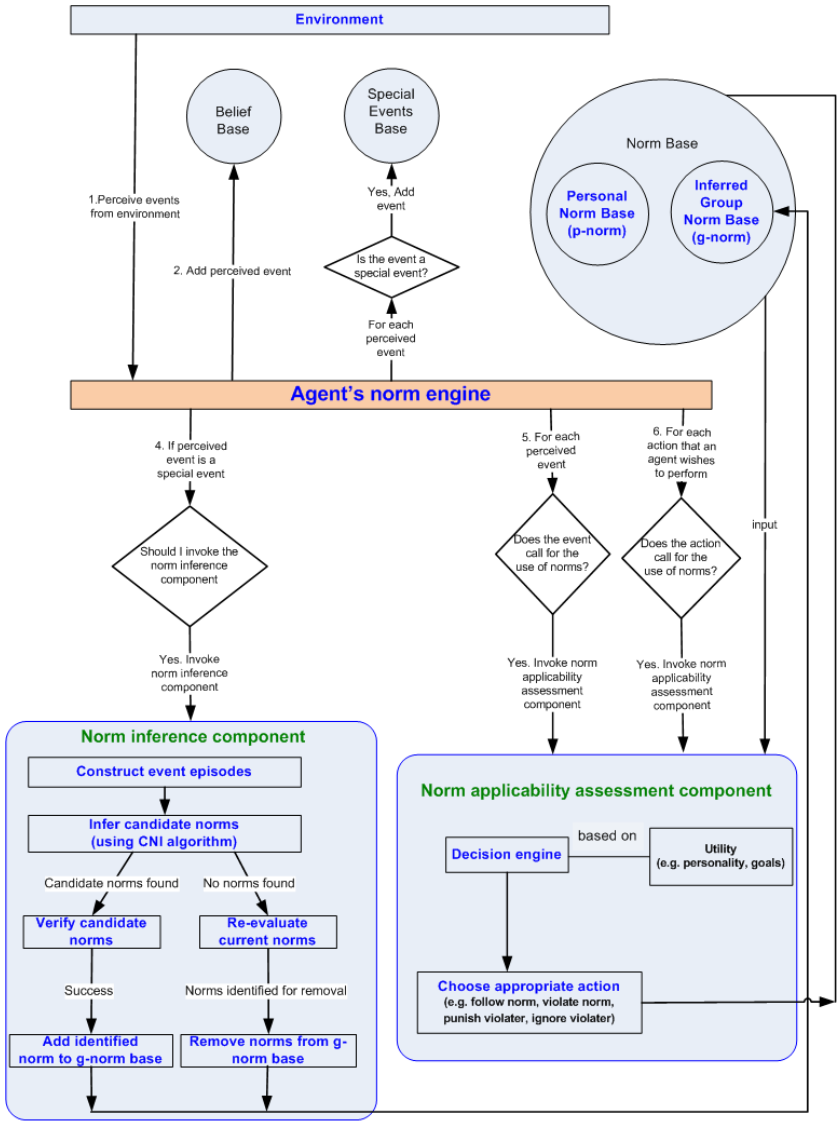

Figure 1: Architecture of the norm identification framework of an agent

component may result in the identification of a *g-norm*, in which case it is added to the *g-norm* base.

An agent, being an autonomous entity, can also decide not to invoke its norm inference component for every occurrence of a special event but may decide to invoke it periodically. When it invokes the norm inference component, it may find a new *g-norm* which it adds to its *g-norm* base. If it does not find

a *g-norm*, the agent may change some of its norm inference parameters and repeat the process again in order to find a *g-norm* or may wait to collect more information.

At regular intervals of time an agent re-evaluates the g-norms it currently has, to check whether those norms hold. When it finds that a *g-norm* does not apply (e.g. if it does not find any evidence of sanctions), it deletes the norm from the *g-norm* base. The operational details of the norm inference component are explained in Section 4.2.

Steps 5: When an event occurs, an agent checks if that event calls for the application of a norm that it may have, either in its *p-norm* or *g-norm* bases. An agent invokes its norm applicability assessment component to decide whether a norm applies and what action it should take when the perceived event is in breach of a norm. This decision may depend upon several factors, such as an agent's utility (e.g. personality and goals). When an agent observes that a norm has been violated, it may decide either to sanction or to ignore the violation based on its utility.

Step 6: When an agent intends to perform a particular action, it may invoke the norm applicability assessment component to check if the action violates its norms (p- and g-norms). When making a decision about norms, an agent may be selfish and may want to maximize its utility and may not follow the norm in the absence of norm enforcers. Another agent may be altruistic and may even bear the cost of punishing another agent that violates the norm. Some agents may be opportunistic norm followers, and they may violate a norm in certain situations. Based on its decision-making factors, an agent can decide whether to perform the action or refrain from the action.

The focus of the rest of the paper is on the norm inference component. The next section describes in details steps 1 to 4 described in this section.

4. Components of the framework

This section describes the components of the norm inference mechanism based on the observation of interaction between agents. The components that will be discussed are a) event storage components and b)the norm inference component. We will describe the role of the components in the context of a park scenario.

4.1. Event storage components

Let us assume that an agent is situated in a public park. The agents are aware that they are in a park, and interactions happen within the park context. Additionally each agent also knows other related information about the environment, such as the location of the rubbish bins. Let us also assume that a norm does not exist to start with but a few of the agents have a notion of what an appropriate action should be in a particular circumstance (a *p-norm*). In this architecture an agent would first observe the interactions that occur between the agents in the society. The interactions could be of two types. The first

type of interaction is the one in which the agent itself is involved and is called a *personnel interaction* (an action that an agent does in an environment or a message that is exchanged with another agent). The second type of interaction is an interaction between other agents that is observed by an observer agent, referred to as an *observed interaction*. The agent records these interactions (events) in its belief base. An agent in the society can assume one or more of the three roles: a participant (P) that is involved in a personal interaction, an observer (O) and a signaller (S).

When the agents move around the park and enjoy the environment, they may become hungry and eat food. Some agents may litter (i.e. drop the rubbish on the ground), and some agents may drop the litter in a rubbish bin. The actions that can be performed by an agent are *move*, *eat* and *litter*. Some agents consider littering to be an activity that should be discouraged, so they choose to signal other agents through actions such as *yelling* and *shaking* their heads in disapproval. We assume that an agent has a filtering mechanism which categorizes actions such as *yell* and *shake-head* as signalling actions. Signalling events can either be positive (e.g. rewards) or negative (sanctions)[2]. These signalling actions are stored in the special events base. The signalling agents can be considered as norm proposers[3]

Let us assume that the agents can observe each other within a certain visibility threshold (e.g. agents can only see other agents in a three cell neighbourhood in a grid environment). An observer records another agent's actions until it disappears from its vicinity. When such an agent observes the occurence of an event that is in breach of one of its norms, the agent may become emotionally charged and perform a sanctioning action, such as shaking its head vigorously in disapproval. Hence, an agent observing this can infer that someone involved in an interaction may have violated a norm. Even though an observer may know that a sanctioning event has occurred, it may not know the exact reason for sanctioning (i.e. it may not know the norm). It will infer norms using the norm inference mechanism.

In order to understand what an agent stores, let us assume that an agent perceives other agents' actions. An event that is perceived consists of an event index, an observed action, and the agent(s) participating in that event. For example an agent observing another agent eating will have the representation of *do(1,eat,A)*. This implies the observer believes that the first event was generated by agent A which performs an action *eat*. A sample representation of events observed by an agent is given below.

[2]In this work we focus on the negative signals (i.e. sanctions)

[3]A norm proposer is an agent that comes up with a norm and recommends the norm to other agents. These norms when created are the "proposed norms". A proposed norm may be created by a designer, leader or entrepreuneur [25]. A proposer may or may not sanction other agents for not following the norms. An authoritative leader may choose to sanction agents that do not follow the proposed norm. A norm entrepreneur may choose to sanction the members of his group for not following the norm. In the off-line design approach there may be penalties for not following the norm.

$$\begin{pmatrix} do(1, eat, A) \\ do(2, litter, A) \\ do(3, move, B) \\ do(4, move, A) \\ do(5, sanction, B, A) \end{pmatrix} (1)$$

An agent records these events in its belief base. Event 5 is a sanctioning event, where agent B sanctions agent A. The agents have a filtering mechanism, which identifies signalling events and stores it in the special events base. It should be noted that special events, such as *yell* and *disapproval shake*, are categorized by an agent as sanctioning events and they are stored in the special events base under the *sanction* event[4].

4.2. Norm inference component

An agent may choose to invoke its norm inference component based on its preference. For example, it can invoke the component every time it perceives a signalling action, or it may invoke this component periodically.

The norm inference component of an agent is made up of two sub-components. The first sub-component makes use of the Candidate Norm Inference (CNI) algorithm to generate candidate norms. Candidate norms are the norms that an agent considers to be potential candidates to become the norms in a society. The second sub-component is the norm verification component, which verifies whether a candidate norm can be identified as a norm in the society.

This sub-section is organized as follows. Firstly we explain the parameters of the CNI algorithm. Secondly we describe the internal details of the CNI algorithm using the park littering example.

4.2.1. Definitions of parameters used in the algorithm

The parameters that are used in the Candidate Norm Inference algorithm are explained below.

History Length (HL): An agent keeps history of the observed interactions for certain window of time. This period of time is represented by the History length (HL) parameter. For example, if HL is set to 20, an agent will keep the last 20 events it observes in it its memory.

Event Sequences (ES): An event sequence is the record of actions that an agent observes in the history. For example the event sequence observed by an agent where HL=5 is given in expression (1).

Special Events Set (SES) : An agent has a set of events it identifies to be special. These events are the signalling events. For example, the special

[4]Recognizing and categorizing a sanctioning event is a difficult problem. In our architecture we assume such a mechanism exists (e.g. based on an agent's past experience)

event set can contain events such as yell, or nod in disapproval (SES = { yell, disapproval nod }). An agent also has the capability to categorize events into two types, sanctions and rewards. For example the actions mentioned above can be identified as sanctioning actions.

Unique Events Set (UES): This set contains the number of distinct events that occur within a period of time. For example, a unique events set for the example given in expression (1) contains[5] the following events, UES = { eat, litter, move, sanction }.

Event Pruning Threshold (EPT): When an agent observes events, not all events may be relevant for identifying potential norms. For example, in a park scenario, an agent's *move* action which occurs frequently may not be relevant to a norm. To exclude these events from the norm inference mechanism, an agent has the EPT threshold. For example if the threshold is set to 0.5 (in a scale of 0 to 1), the agent will exclude all events that have the Occurrence Probability (OP) which is greater than 0.5. Occurrence Probability of event E is given by the following formula.

$$OP(E) = \textit{(Number of occurrences of E) / (Total number of events in ES)}$$

In the example shown in expression (1), OP(eat) = 0.2, OP(litter) = 0.2, OP(move)=0.4 and OP(sanction)=0.2. An agent can choose to increase or decrease the EPT threshold in order to find a norm.

Window size (WS): When an agent wants to infer norms, it looks into its history, a certain number of recent events that precede a sanction. For example, if the WS is set to 3, an agent constructs an Event Episode (EE) with three events that precede a special event. Construction of event episodes is described in the next sub-section. It should be noted that an EE is a subset of ES.

Norm Identification Threshold (NIT): When coming up with candidate norms, an agent may not be interested in events that have a lower probability of being a norm. For example, if an agent sets NIT to be 50 (in a scale from 0 to 100), it indicates it is interested to find all sub-episodes of an event episode that have 50% chance of being a candidate norm (i.e. being the reason for generating a sanction).

Norm Inference Frequency (NIF): An agent may choose to invoke a norm inference component every time it observes a special event, or may invoke the component periodically. An agent has a parameter called norm inference frequency (NIF) that specifies what the time interval between two invocations of the norm inference component are. An agent, being an autonomous entity, can change this parameter dynamically. If it sees that the norm in a society is not changing, then it can increase the waiting period for the invocation of the norm inference component. Alternatively, it can reduce the time interval if it sees the norm is changing.

[5] Assume that event occurrences can be modelled as simple propositions

4.2.2. Candidate Norm Inference (CNI) algorithm

There are three main steps involved in the Candidate Norm Inference algorithm (see algorithm 1). First, event episodes are created. Second, commonly occurring events are pruned. Third, the Candidate Norm List (CNL) is generated using a modified version of the WINEPI algorithm [28].

Algorithm 1: Candidate Norm Inference algorithm (main algorithm)

```
1 foreach invocation of the norm inference component do
2 |   Create event episodes ;                    /* see Algorithm 2 */
3 |   Prune event episodes ;                      /* see Algorithm 3 */
4 |   Create Candidate Norm List (CNL) ;          /* see Algorithm 4 */
5 end
```

Creating event episodes

An agent records other agents' actions in its belief base. We call these events that were recorded in the belief base as event sequences (ES). An agent has a certain history length (HL). Let us assume that there are three agents A, B and C as given in expression (1). Agent A eats, litters and moves, while agent B moves and then sanctions. Agent C observes these events and categorizes them based on which agent was responsible for creating an event. The set $\{A\}$ followed by a right arrow ($\rightarrow$) indicates the categorization of events performed by agent A as observed by agent C. A hyphen separates one event from the next.

$$\{A\} \rightarrow do(1, eat, A) - do(2, litter, A) - do(4, move, A)$$
$$\{B\} \rightarrow do(3, move, B) - do(5, sanction, B, A)$$

When an agent observes a special event (e.g. sanction), it extracts the sequence of actions from the recorded history (event sequences (ES)) that were exchanged between the sanctioning agent and the sanctioned agent. In the example shown above, the observer infers that something that agent A did may have caused the sanction. It could also be something that agent A failed to do might have caused a sanction. In this work we concentrate on the former. Agent C then extracts the following sequence of events that took place between A and B based on the information retrieved from its history. We call the retrieved event sequence that precedes a sanction as the event episode (EE).

$$\{A, B\} \rightarrow do(1, eat, A) - do(2, litter, A) - do(4, move, A) - do(5, sanction, B, A)$$

To simplify the notation here, only the first letter of each event will be mentioned from here on(e.g. e for eat). Thus the event episode for interactions between agents A and B shown above will be represented as

$$\left(\ \{A, B\} \rightarrow e - l - m - s \ \right)$$

There might be a few sanctioning events at any given point of time that an agent observes. A sample event episode list (EEL) that contains events that are observed by an agent preceding an action where WS=3 is given below.

$$\left(\begin{array}{l} e-l-m-s, l-e-l-s, m-e-l-s, e-l-e-s, e-l-e-s \\ l-e-l-s, e-e-l-s, m-e-l-s, e-l-m-s, e-l-e-s \end{array} \right) (2)$$

The pseudocode for creating an event episode list (EEL) is given in Algorithm 2. For every special event in the event sequence (ES), an agent creates an event episode (EE) with n events that precede the special event where, n=WS[6]. EE is then added to the event episode list (EEL).

Algorithm 2: Pseudocode to create Event Episode List

```
Input: Event Sequence (ES), Window Size (WS)
Output: Event Episode List (EEL)
foreach invocation of the norm inference component do
    foreach special event in ES do
        Create an Event Episode (EE) with the last n events that precede
        the special event where n=WS;
        Store EE in EEL;
    end
end
```

Pruning event episodes

Once an agent creates the EEL, it will prune the EEL. The objective of pruning is to eliminate those events that may not be relevant to the norm. For example, in the park scenario, an agent may observe that the action *move* happens 75% of the time. An agent can assume that commonly occurring events, such as *move*, may not be a reason for the sanction. So the agent prunes those commonly occurring events from its EEL. Commonly occurring events are identified using the occurrence probability (OP) of the event in the event sequence (ES). If the occurrence probability of an event in ES is greater than or equal to the norm pruning threshold (NPT), then the event will be removed from the EEL. Hence, the pruned event episode list (PEEL) will only contain events that do not occur frequently. An agent can vary the NPT depending upon its needs (e.g. if an agent does not find a norm, it can decrease its NPT). Pruning reduces the search space of an agent in identifying candidate norms.

The pseudocode for creating the pruned event episode list (PEEL) is given in Algorithm 3. First, an unique event set (UES) is created. For every event in the event sequence (ES), if the event does not already exist in the UES and if the event is not a special events set (SES) then it is added to the unique event

[6]In this example, we have assumed that an agent considers three events (n=3) that precede a signal (a sanction). The value of n can change and an agent being a computation machine should be able to handle a large number of possible events.

set (lines 1 to 5). Second, events that should be pruned are identified and stored in the removable events set (RES). If the occurrence probabilities of an event in the UES is greater than NPT, it is added to the RES (lines 6 to 11). Third, the events that are present in the removable event set are pruned from the EEL (lines 12 to 18).

Algorithm 3: Pseudocode to create Pruned Event Episode List

```
   Input: Event Episode List (EEL), Unique Event Set (UES), Norm
          Pruning Threshold (NPT)
   Output: Pruned Event Episode List (PEEL)
   /* construct Unique Event Set (UES)                          */
 1 foreach event E in ES do
 2 |  if E ∉ Unique Event Set (UES) and E ∉ Special Event Set (SES)
   |  then
 3 |  |  Add E to UES;
 4 |  end
 5 end
   /* construct Removable Event Set (RES)                       */
 6 foreach event E in Unique Event Set (UES) do
 7 |  Calculate OP(E);
 8 |  if OP(E) ≥ NPT then
 9 |  |  Add event to Removable Event Set (RES);
10 |  end
11 end
   /* construct Pruned Event Episode List (PEEL)                */
12 foreach Event Episode (EE) in EEL do
13 |  foreach event E in an EE do
14 |  |  if event E in RES then
15 |  |  |  Remove event from EE;
16 |  |  end
17 |  end
18 end
```

Assuming that NPT is set to 50 and the $OP(m)$ is greater than 50 in an ES, then the PEEL corresponding to the EEL shown in expression (2) is given below.

$$\begin{pmatrix} e-l-s, l-e-l-s, e-l-s, e-l-e-s, e-l-e-s \\ l-e-l-s, e-e-l-s, e-l-s, e-l-s, e-l-e-s \end{pmatrix} (3)$$

Generating candidate norms list (CNL)

The pseudocode for generating the Candidate Norms List (CNL) is given by Algorithm 4. Algorithm 4 is a modified version of the WINEPI algorithm [28], an association rule mining algorithm[7]. The WINEPI algorithm analyses event

[7]Association rule mining [27] is one of the well known fields of data mining where relation-

sequences and identifies frequently occurring episodes in a particular window of time.

As norms in this work are sequences of events that precede a signalling event, we have used a modified version of the WINEPI algorithm to identify candidate norms. For example littering action may happen 100% of the time before the occurrence a sanctioning event. In this case, the WINEPI algorithm can be used to identify the relationship between the littering action and the sanctioning action. We have modified the WINEPI algorithm such that it generates sub-episodes using "permutation with repetition". The pseudo code of the modification is given in algorithm 5.

Algorithm 4 works in several iterations. The number of iterations is equal to Window Size (WS). The sub-episodes for the first iteration are of length one. The sub episode list (SEL) for iteration one contains all the events in the UES. For example, for the events listed in expression (3), the SEL at the first iteration will contain events e and l. For each of the sub-episodes in SEL, the occurrence probabilities are calculated. If the occurrence probability of a sub-episode in the pruned event episode list is greater than or equal to the norm inference threshold (NIT), the event is added to the Candidate Norms List *(lines 7 to 11)*. For example, if the occurrence probabilities of events e and l are greater than or equal to NIT, then these will be added to CNL. Each candidate norm is also added to a temporary list which is used for creating the SEL for the next iteration. The SEL for the next iteration (SEL_{next}) is created using Algorithm 5 and is assigned to ($SEL_{current}$).

Each sub-episode in the second iteration will have two events. In the second iteration a sub-episode in SEL will be added to CNL if two conditions are satisfied (lines 12 to 16).

- Each event in the sub-episode should already exist in the CNL.
- The occurrence probability of the sub-episode in the pruned event episode list (PEEL) should be greater than or equal to NIT.

In a similar fashion, the algorithm computes all candidate norms. The maximum length of a candidate norm (and the number of iterations) is equal to WS.

---

ships between items in a database are discovered. For example, interesting rules such as 80% of people who bought diapers also bought beers can be identified from a database. There are several well known algorithms that can be used to mine interesting rules such as Apriori [29] and WINEPI [28]

Algorithm 4: Pseudocode to create Candidate Norms List (CNL)

```
Input: Pruned Event Episode List (PEEL), Unique Event Set (UES),
       Window Size(WS), Norm Inference Threshold(NIT)
Output: Candidate Norms List (CNL)
begin
    iterNum = 1;
    Sub-Episode List (SEL_current) = UES
    while iterNum ≤ WS do
        SEL_temp =∅;
        foreach Sub-Episode(SE) in SEL that appears in PEEL do
            if iterNum = 1 then
                if OP(SE) ≥ NIT then
                    Add SE to CNL, Add SE to SEL_temp;
                end
            end
            else
                if each event in SE ∈ CNL and OP(SE) ≥ NIT then
                    Add SE to CNL, Add SE to SEL_temp;
                end
            end
        end
        Construct SEL_next using SEL_temp ;        /* algorithm 5 */
        if iterNum < WS then
            iterNum = iterNum + 1;
            SEL_current = SEL_next;
        end
        else return CNL;
    end
end
```

Algorithm 5: Pseudocode to create Sub-Episode List

```
Input: Candidate norms list (tempList)
Output: Sub Episode List (SEL)
foreach candidate norm in tempList do
    Generate Sub-Episodes (SE) of length n using other candidate norms
    (allowing repetition of events);
    Add each generated SE to SEL;
end
```

Let us assume that an agent is interested in *three* events in an event sequence that precede a sanction (i.e. event episodes of length three). Let us assume that NIT is set to 50%, and the unique event set is {e,l}. As an example let us consider the pruned event episode list (PEEL) given in expression (3). In the first iteration, $SEL_{current}$ contains sub-episodes of length one which are e and

l. For each sub-episode an agent calculates the occurrence probability. The occurrence probabilities for both sub-episodes in this case are 100%. So, the agent adds both of these sub-episodes to its candidate norm list. For the second iteration, the agent has to calculate sub-episodes of length two that have NIT greater than 50%. It uses Algorithm 5 to calculate the next sub-episode list (SEL_{next}). For this purpose it uses the candidate norms that were found in the previous iteration (SEL_{temp} in algorithm 4).

Algorithm 5 creates sub-episodes for the subsequent iteration based on the candidate norms from the previous iteration. In the running example, in the second iteration the algorithm creates sub-episodes of length two based on sub-episodes of length one. Allowing repetition of events, the algorithm creates the following sub-episodes {*ee, el, le, ll*} and adds them to the SEL_{next}[8]. Then, the probabilities of these 4 sub-episodes are calculated. Occurrence probabilities of { *ee, el, le, ll* } are { 10%, 100%, 50%, 0% }. As NIT is set to 50%, *el* and *le* are added to the candidate norms list. These two sub-episodes will be considered for the creation of SEL for the third iteration using Algorithm 5. For the third iteration the contents of the SEL are {*ele, lel*}.

In iteration three, the occurrence probabilities of {*lel,ele*} are {20%, 30%}. As the occurrence probabilities of the sub-episodes of length three are below NIT these events will not be added to the candidate norms list. As the number of iterations is equal to WS, the algorithm returns the candidate norm list to the agent. In the end, the candidate norms list will have the following entries whose occurrence probabilities are greater than or equal to NIT: {*e,l,el,le*}. If the agent sets the NIT to 100% then the CNL will contain *e*, *l* and *el*.

It should be noted that algorithm 4 is a modified version[9] of the WINEPI algorithm [28]. The modification is given in Algorithm 5 can identify candidate norms that are obtained by considering "permutations with repetition" when constructing sub-episodes. We note that algorithm 4 can be replaced with other association rule mining algorithms. Hence, it forms the replaceable component of the CNI algorithm.

Having compiled a set containing candidate norms, the agent passes this information to the norm verification and identification component.

4.2.3. Norm verification and identification

In order to find whether a candidate norm is a norm of the society, the agent asks another agent in its proximity. This happens periodically (e.g. once

[8]Permutations with repetitions are considered because an agent does not know whether littering once (*l*) is a reason for sanction or littering twice (*ll*) is the reason for the sanction. It could be that littering once may be allowed but an agent littering twice may be punished

[9]Some well known algorithms in the data mining field can be used for mining frequently occurring episodes (i.e mining association rules) [29, 28]. A limitation of the well-known Apriori [29] algorithm is that it considers combinations of events but not permutations (e.g. it does not distinguish between event sequences *el* and *le*). WINEPI [28] addresses this issue, but it lacks support for identifying sequences that are resultants of permutations with repetition (e.g. from sub-episodes of length one, e.g. *e* and *l*, the algorithm can generate sub-episodes of length two which are *el* and *le*, but not *ee* and *ll*).

in every 10 iterations).

When two agents A and B interact, A chooses its first candidate norm (say *el*) and asks B if it knows whether *el* is a norm of the society. If the response is affirmative, A stores this norm in its set of *identified norms*. If not, A moves on to the second candidate norm in its list[10].

In the case of the running example, the sub-episode e has the highest probability for selection and it is chosen to be communicated to the other agent. It asks another agent (e.g. an agent who is the closest) whether it thinks that the given candidate norm is a norm of the society. If it responds positively, the agent infers $prohibit(e)$ to be a norm. If the response is negative, this norm is stored in the bottom of the candidate norm list. It then asks whether l is the reason for sanction. If yes, littering is considered to be prohibited. Otherwise, the next event in the candidate norm list is chosen. This process continues until a norm is found or no norm is found, in which case the process is re-iterated once a new signal indicating a sanction is generated. When one of the candidate norms has been identified as a norm of the society, the agent still iterates through the candidate norm list to find any co-existing norms.

Note that an agent will have two sets of norms: candidate norms and identified norms. Figure 2 shows these two sets of norms. Once an agent identifies the norms of the system and finds that the norms identified have been stable for a certain period of time, it can forgo using the norm inference component for a certain amount of time (based on the norm inference frequency (NIF)). It invokes the norm inference component periodically to check if the norms of the society have changed, in which case it replaces the norms in the identified list with the new ones (or deletes the norms which are no more applicable).

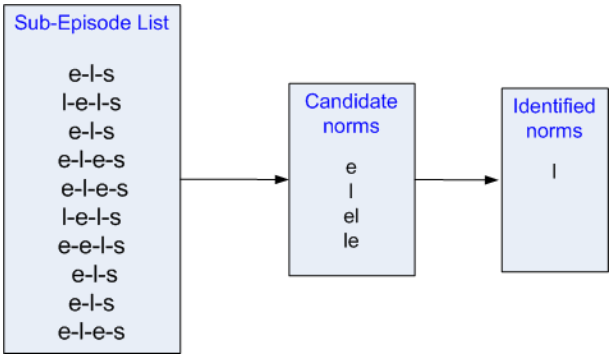

Figure 2: Two sets of norms

[10]Other alternative mechanisms are also possible. For example, an agent could ask for all the candidate norms from another agent and can compare them locally.

4.3. Norm applicability assessment component

An agent invokes the norm applicability assessment component under two circumstances. Firstly, when an event occurs it should check if it violates its p-norms or g-norms. An agent invokes its norm applicability assessment component to decide whether a norm applies and what action it should take when the perceived event is against the norm. This decision may depend upon several factors such as an agent's utility (e.g. personality and goals). When an agent observes that a norm has been violated, it may decide either to sanction or to ignore the violation based on its utility. Secondly an agent may invoke this component if it intends to perform an action. The agent will check if its intended action violates its norms (p- and g-norms). When making a decision about norms, an agent may be selfish and may want to maximize its utility and may not follow the norm in the absence of norm enforcers. Another agent may be altruistic and may even bear the cost of punishing another agent that violates the norm. Some agents may be opportunistic norm followers and they may violate a norm in certain situations. Based on its decision making factors an agent can decide whether to perform the action or refrain from the action.

A detailed discussion of the norm applicability assessment component is outside the scope of this paper, as the main focus of the paper is on norm identification (the rounded rectangle in lower left part of Figure 1).

5. Experiments on norm identification

In this section we demonstrate how the agents that make use of the proposed architecture are able to infer the norms of the society.

We have implemented a Java-based simulation environment to demonstrate how norms can be identified by agents in a multi-agent society. A toroidal grid represents a social space where agents can move. An agent enjoys the park by moving from one location to another. An agent can move in one of the four directions (up, down, left and right). There are three types of agents in the system: the learning litterers (LL), the non-litterers (NL) and the non-littering punishers (NLP). An agent's visibility is limited to a particular zone. The snapshot given in Figure 3 shows different types of agents in a grid environment situated in four different societies. The agents are represented as circles. The letters that appear above an agent specify the agent number and the action it is currently performing. When an agent infers a norm, a solid square appears inside the circle with the same colour as that of the signalling agent. The signalling agent is a norm proposer which punishes other agents probabilistically based on its *p-norm*. For experiments 1 to 5, all the agents make use of the norm inference mechanism.

5.1. Experiment 1 - Norm identification and verification

The objective of this experiment is to demonstrate that agents that use the norm inference architecture can generate candidate norms and also identify norms through the verification process. Agents in a society can *verify* that

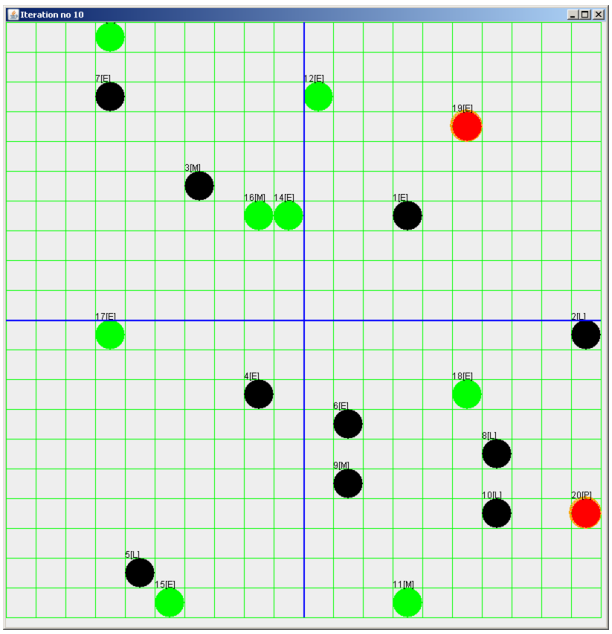

Figure 3: Snapshot of the simulation environment

certain norm holds in a society by asking other agents in the society. There were 100 agents in the agent society (50 NL, 46 LL and four NLP agents). The NLP agents punished agents that littered.

5.1.1. Norm identification

In order to demonstrate that the norm identification component works, we conducted experiments by varying the NIT and keeping all the other parameters constant. For a particular agent, when NIT was set to 25%, the agent inferred 7 candidate norms {e,el,l,ee,le,lel,eee} whose occurrence probabilities were {1,0.5,0.5,0.41,0.25,0.25,0.25}. When NIT was set to 50%, the agent inferred 3 candidate norms {e,el,l} whose occurrence probabilities were {1,0.5,0.5}. Note that the candidate norms that are identified when NIT was set to 50% is a sub-set of the candidate norms that were identified when NIT was set to 25%. When NIT was set to 100%, the agent inferred only one candidate norm {e}.

5.1.2. Norm verification

In our experimental set-up an agent can ask one other agent in its vicinity(randomly chosen) about a candidate norm. If that agents answers positively, then the agent will promote the norm to the identified norm list.

In our experimental set-up, when seeking norm verifications, an agent can use one of the following approaches. It can either ask a) a sanctioning agent or b) any agent that possesses a norm (e.g. the agent may have obtained the norm from a sanctioning agent). As the probability of the other agent being

a non-sanctioning agent is higher than being a punishing agent (0.96 vs. 0.04), the norm identification is faster (see Figure 4) when any agent that has norm can recommend the norm to other agents (approach b) as opposed to norm recommendation only by the sanctioning agents (approach a).

This simple experiment reflects what happens in human societies. An agent entering a new society asks other agents in a society whether a suspected norm currently holds in a society. The experiments that are reported in the rest of the paper use norm recommendations from any agent.

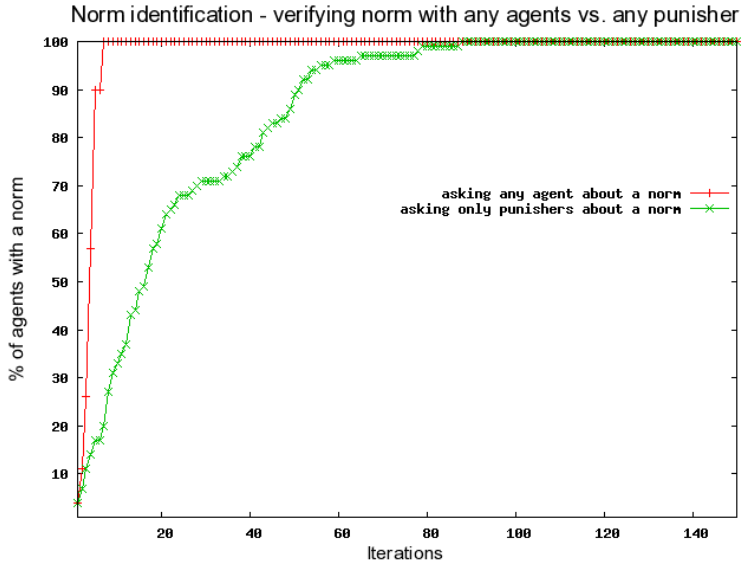

Figure 4: Norm verification - enquiring about a norm to a punisher vs. any agent in the society

We also conducted experiments with two types of punishers in the society, one that punishes eating activity and the other that punishes littering. The results were similar to the one shown in Figure 4.

5.2. Experiment 2 - Dynamic norm change (from society's view point)

Agents in a society should have the ability to infer norms when norms change. A new norm can be established when a punisher leaves the society and a new punisher joins the society or when a punisher agent comes up with a new norm replacing the old one. This experiment demonstrates that agents can change norms based on inference. Figure 5 shows three lines corresponding to agents with a) no norms, b)a norm against eating and c)a norm against littering. There were 100 agents in a society. The sanctioning agents could move into a society and also leave a society. Depending upon the presence of a sanctioning agent the norm of the society changes. It can be observed that around iteration 225, all agents have inferred the norm against eating and around iteration 585, all agents have inferred the norm against littering. It should be noted that after

iteration 585, both the norms coexist in the society (assuming that a norm could be said to have emerged if 75% of the agents possess the norm).

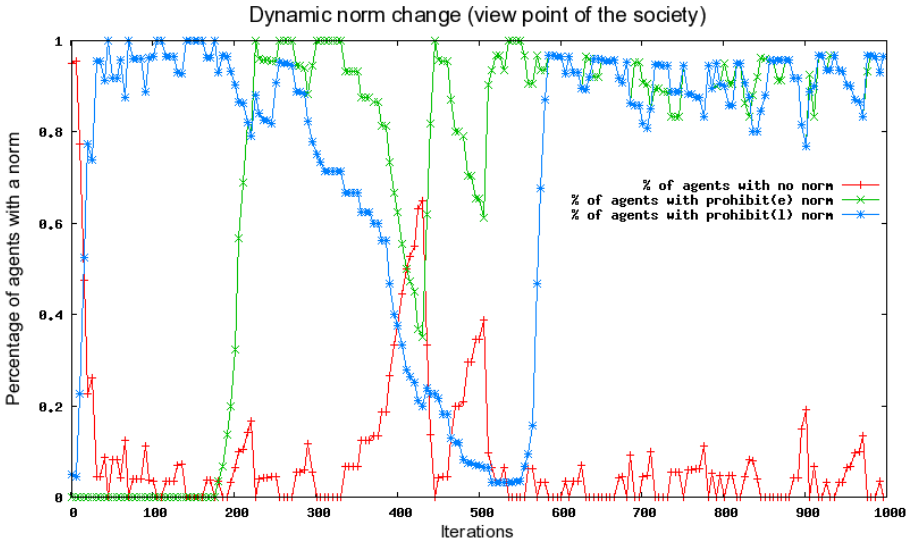

Figure 5: Norm change from the view point of a society

5.3. Experiment 3 - Dynamic norm change (from individual agent's view point)

Let us assume that an agent moves across four different societies which may have different kinds of norms. An agent in our architecture will be able to infer different types of norms (a norm against eating, a norm against littering and a norm against both eating and littering). Figure 6 shows the result of norm inference for an agent. It can be observed that an agent is able to infer the change of norm when it moves from one society to another.

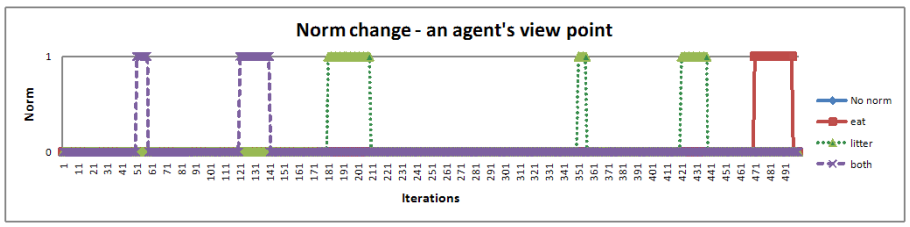

Figure 6: Norm change - an individual agent's point of view

5.4. Experiment 4 - Adaptability of an agent

Norm Identification Frequency (NIF) and Norm Identification Threshold (NIT) are two parameters that an agent can vary based on it its success in

recognizing a norm in the society. Norm Identification Frequency (NIF) refers to how often an agent invokes the norm inference component to obtain a candidate norm list. For example if NIF is set to five, an agent invokes the norm inference components once in five iterations. Norm Identification Threshold (NIT) refers to the threshold that an agent sets in order to obtain a candidate norm list. For example if NIT is set to 50% for an agent, the agent is interested to obtain a candidate norm list where the probability of norm occurrence of a sub-episode is greater than or equal to 50%.

An agent in our set-up starts with NIF=5. When it does not find any norm it retains the same NIF value. Once it has found all the norms in the society (i.e. there are no new norms to be found after a certain number of iterations), it increases its NIF value by a step function (an increase of 5 in our case). The new value of NIF will be 10 which implies that the agent will infer norms only once in 10 iterations. This continues till a norm changes (i.e. a new norm is found). When a new norm is found, the NIF is set to back to 5. Figure 7 shows how the NIF changes by keeping NIT to be constant (50% in our case). It should be noted that when an agent moves into a new society, the NIF is set to 5.

Figure 7 shows two lines corresponding to an agent moving in one society and within four societies with static punishers. All these societies have only one type of punisher (i.e. only one type of norm). It can be observed that when the agent moves within one society, it infers the norm, and hence its NIF value increases (as the norm does not change). In the case of the agent moving in four different societies, the agent's NIF increases after it has found a norm, as long as it is in the same society where it found the norm. When the agent moves to a new society[11], the agent's NIF is set to the base value, and then it starts increasing once it has found the norm again. The "sudden jumps in values (that resemble vertical lines)" that occur in regular intervals indicate that an agent has moved from one society to another. It should be noted that when the punishers are moving within the societies (not shown in the figure), the NIF values (i.e. the NIF line) of the agent is different from when the punishers were static. This is because when the punishers move from society 1 to 2, agents in society 1 are not able to recognize the norm when they infer the norm the next time (as there is no evidence of sanctions).

An agent in our set-up starts with NIT=50%. When an agent invokes the norm inference component (NIF value is met), it initially uses the default NIT value. When the norms do not change the agent increases the NIT value by a step function (an increase of 5 in our case). When no norm is found then it decreases the NIT value by a step function (a decrease of 5 in our case). An agent increases its NIT because it can reduce extra computation that is needed to infer a candidate norm. For example if the same set of norms are obtained when an agent has NIT=100%, there is no reason why an agent should retain

[11]We assume that the agent knows when it enters a new society. For example, the agent may know the physical boundaries of the society in which it is situated

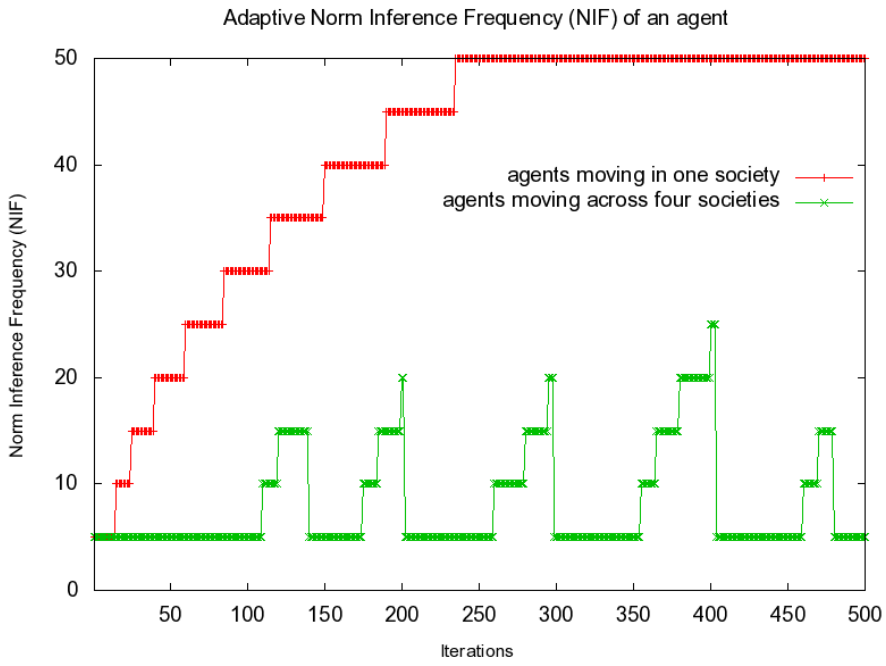

Figure 7: Norm verification - enquiring about a norm to a punisher vs. any agent in the society

the value of NIT=50% where it has to perform some additional computation to infer the same norm (based on the CNI algorithm). An agent reduces the NIT, because it wants to explore the search space of candidate norms that is below its initial threshold value. Note that when an agent moves into a new society, the NIT is set to 50%. Figure 8 shows how the NIT changes by keeping NIF constant (NIF=5 in our case). The vertical lines that occur periodically indicate that an agent has moved from one society to another. The initial value of NIT is set to 50%, because an agent should not start from scratch to infer a candidate norm (i.e. from NIT=0%) in order to avoid extra computation.

It can be noted from Figure 8 that when an agent moves within one society with one type of norm, its NIT threshold gradually reaches the value of 100, as it is able to infer the norms with a high level of support. In the case of the agent moving across societies, an agent's NIT value starts increasing when the agent has identified the norm, but it drops to 50 when it moves from one society to another. For the same set-up when the punishers are moving, the agent may not be able to infer the norms when the punishers move into a new zone. So, the NIT line showing the movement of an agent with static punishers is different from the scenario involving dynamic punishers (not shown in the figure).

An agent can vary both NIF and NIT. Figure 9 shows how both of these variables change in an agent. The circled region is of interest, because it shows

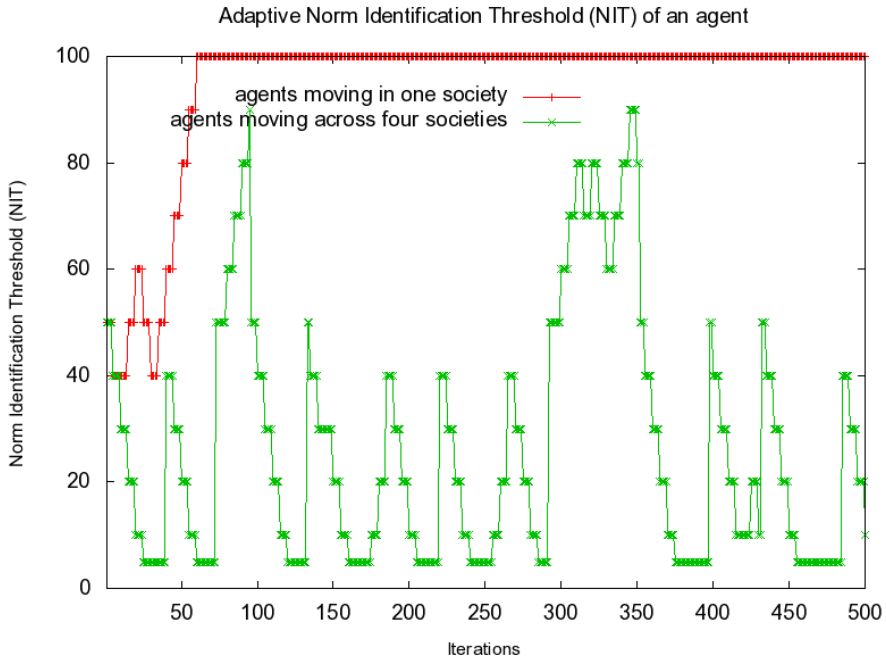

Figure 8: Norm verification - enquiring about a norm to a punisher vs. any agent in the society

that an agent did not infer a norm initially (iteration 80 to 111). So the NIT line drops. It inferred the norm around iterations 111 to 133, which is indicated by the upwards trend. Then the agent did not infer the norm between iterations 134 to 139. Hence there is a drop in the NIF line. In iteration 140, the agent has moved to a new zone. The NIF line is similar to the one shown in Figure 7. The line that appears in the bottom shows when an agent moves from one society to another.

Experiments 3, 4 and 5 have demonstrated that an agent can dynamically change its norm inference behaviour. An agent, being an autonomous entity, can decide when to increase or decrease these parameters.

5.5. Experiment 5 - Using norm history

When an agent moves from one society to another, it can record the norm of the society it is leaving, so that the information can be used when it comes back to the same society. The experimental result shown in Figure 10 demonstrates that the percentage of agents that have identified a particular norm in an agent society is high when agents store the norm (possess history) of the society. The recorded history can be used when an agent re-enters the society that it has previously been to.

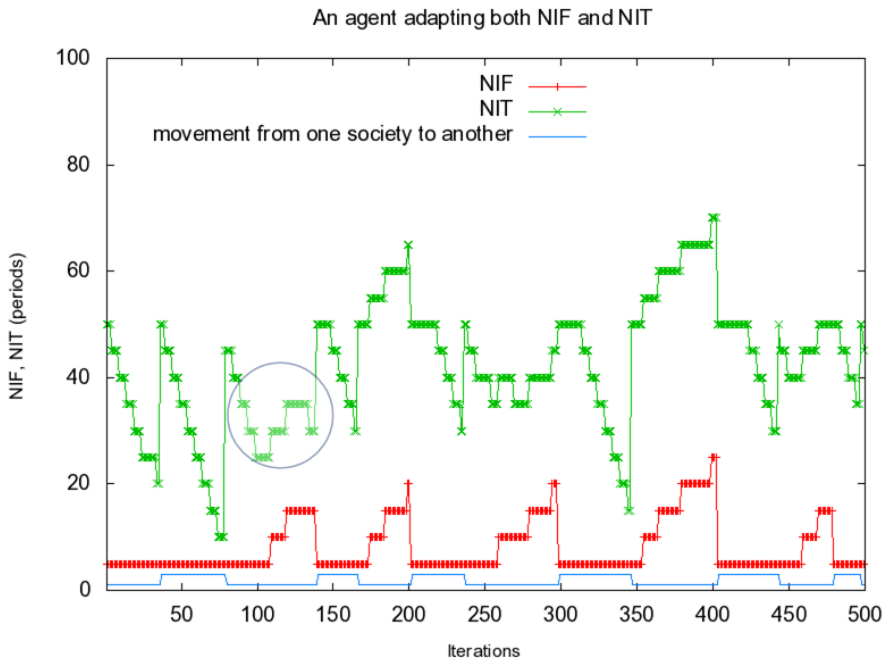

Figure 9: Norm verification - enquiring about a norm to a punisher vs. any agent in the society

Note that researchers have demonstrated agents that record history will be better off than agents that do not have a norm history [30]. The objective of this experiment is to compare how much better the agent will be if it records the norms of the society that it leaves. It can be seen that on average about 80% of the agents can infer norms when history was stored while only 50% of the other agents on average inferred norms (for the same experimental set-up). A higher percentage of agents inferred norms when using their history, because the agents that come in with history information can start asking other agents in the society whether a norm holds (norm verification). If an agent does not have a norm history, it first has to infer the norm (i.e. invoke the norm inference component) and then ask another agent for norm verification, which is slower than using the norms in the norm history at the verification stage.

Using history is useful when the punishers are not moving (i.e. the norms in the society are stable). If the punishers are moving then the norms may change (i.e. when there are different types of punishers). If the norms change then the mechanism may not be very useful. If there are a large number of separate societies, then the agent may not come back to a previously inhabited society. In this case, the history information may not be useful. However, the agent is better off keeping the history if it comes back to a society it has previously been to, since it does not have to start inferring the norms from scratch.

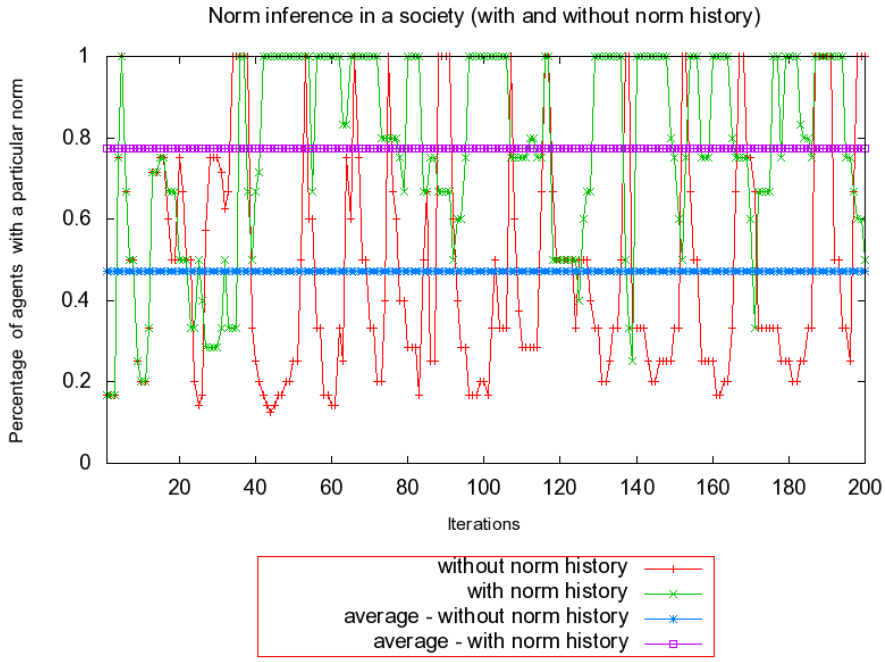

Figure 10: Norm verification - enquiring about a norm to a punisher vs. any agent in the society

6. Experiments based on the utility of an agent

An agent being an autonomous entity, may choose to maximize its utility. Such utilitarian agents may choose to become a part of a society that maximizes their utility. In this section we describe two experiments that we have conducted using utilitarian agents. The objectives of experiments are two-fold.

1. To demonstrate that the utility of a norm-abiding agent is better in a normative society than a society with no norms.
2. To demonstrate that when agents are capable of norm inference, the norm establishment in a society is faster than when agents do not infer norms.

6.1. Experimental set-up

Let us assume that there are two societies: a normative society and a society with no norms. There are three types of agents: learning litterers (LL), non-litterers (NL) and non-littering punishers (NLP) in both the societies. An agent has a utility value which we call the satisfaction level (S) which varies from 0 to 100.

An agent's satisfaction level (S) decreases in the following situations:

- When a non-litterer observes a littering action its satisfaction level decreases (-1).

- When a litterer is punished, its utility decreases (-1).
- For all agents, littering activity results in the decrease of the utility. This is because each littering activity ruins the commons area (-1/number of agents in the society).

An agent's satisfaction level (S) increases (i.e. it gains utility) in the following situations:

- When a litterer litters, it gains utility in a society (+1).
- When a non-litterer does not see any littering action in a society, its utility increases (+1).

At the start of experiments, an agent moves across two societies (society 1 and 2). All the agents start with a satisfaction level of 50. This value can increase or decrease based on the society the agent is in. When the agents move into a new society the satisfaction level is set to 50. When an agent has been to both the societies, an agent chooses the society for which it has a higher utility.

6.2. Experiment 1 - Norm abiding agents are better off in a normative society

Using the experimental set-up described in the previous section, we experimented how the agents organized themselves into two societies based on utility. In one of the societies the punishers were present (society 2). In the other society the punishers were absent. In this experiment, the agents do not make use of norm inference mechanism. They are utility maximizers (i.e. they would move to a society that yields better utility).

At the start of the experiment, the agents moved in both societies. At regular intervals of time (say every 50 iterations) each agent decides which society to choose. The agent evaluates its satisfaction level based on the societies it inhabited. The litterers' utility in society 1 is better than society 2 because in society 2 they are punished. So they move towards society 1. As the litterers move towards society 1, the non-litterers move towards society 2 because their utility in that society is better due to the absence of litterers. It can be observed in Figure 11 that at the end 160 iterations, the litterers have moved from society 2 to society 1. When there are litterers in society 1, the non-litterers move to society 2. It can be observed that at the end of 350 iterations all the non-litterers have moved into society 2.

We also conducted experiments by making the punishers move across societies at certain intervals of time. In this set-up, we ran the experiments for 1500 iterations. In the first 500 iterations, the punishers were in society 2, and in the second 500 iterations the punishers moved to society 1. In the third 500 iterations the punishers moved back to society 2.

In the first 500 iterations, the agents were separated into two groups as described above. After 500 iterations, when the punishers move to society 1, the utility of litterers in that society starts decreasing, so, the litterers move to society 2. When they move to society 2, the non-litterers' utility decreases,

so they move to society 1. Again it was observed that the two societies were separated based on the personality of the agents (society 1 with non-litterers and society 2 with litterers). After 1000 iterations when the punishers have moved to society 2, the same process continues. Society 1 now has all the litterers and society 2 has non-litterers.

This experiment demonstrates that norm-abiding agents are better off in the normative society where the norm violation is punished by sanctioning agents, and the non-litterers are better off to be in a society with no norms. This experiment also demonstrates that a normative agent is adaptive, as it moves from one society to other if its utility decreases in the society.

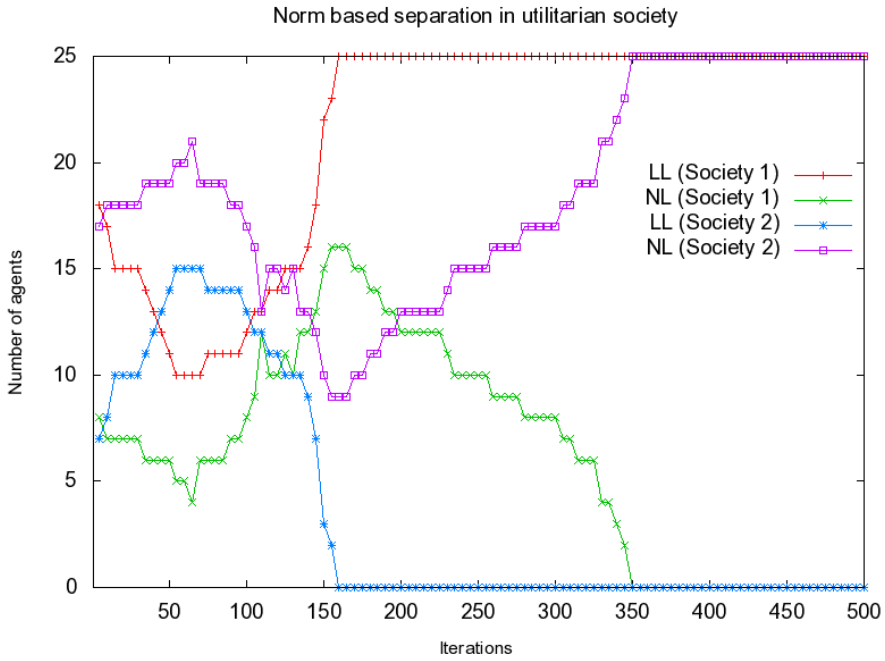

Figure 11: Separation of agents into two groups based on utility

6.3. Experiment 2 - Utilitarian vs. Hybrid strategies for agents

In the previous experiment the agents employed the utilitarian strategy. In this experiment the agents use the norm inference mechanism. At the same time they also computed utility. We call this strategy a hybrid strategy. Except for the change of the strategy the experimental set up was similar to the previous experiment. We observed that the overall separation of the two groups is faster when the agents are able to apply the norm-inference mechanism along with the utilitarian mechanism. This is because when the litterers in society 2 infer that there is a norm against littering, then they will decide not to litter in the society (to minimize the decrease in their utility) and also decide to move to

another society. When a non-litterer in society 2 infers that there is a norm then it decides to stay in that society as it knows that it would be better off in this society (as the punishers will punish the littering agents). So, the separation of agents into two societies is faster when a norm inference mechanism is used along with the norm inference mechanism.

Figure 12 shows the number of litterers and non-litterers in two societies with the two types of strategies for the non-littering agents. It can be noted that when the hybrid strategy was employed by the agents, the separation of agents into two separate groups was faster than when the agents used just the utilitarian strategy. The hybrid strategy resulted in the littering agent moving from society 2 to society 1 faster than when using utilitarian society. In this experiment, the system using hybrid strategy converged 100 iterations before the system that used utilitarian strategy. As the litterers moved to society 2, the non-litterers moved to society 1.

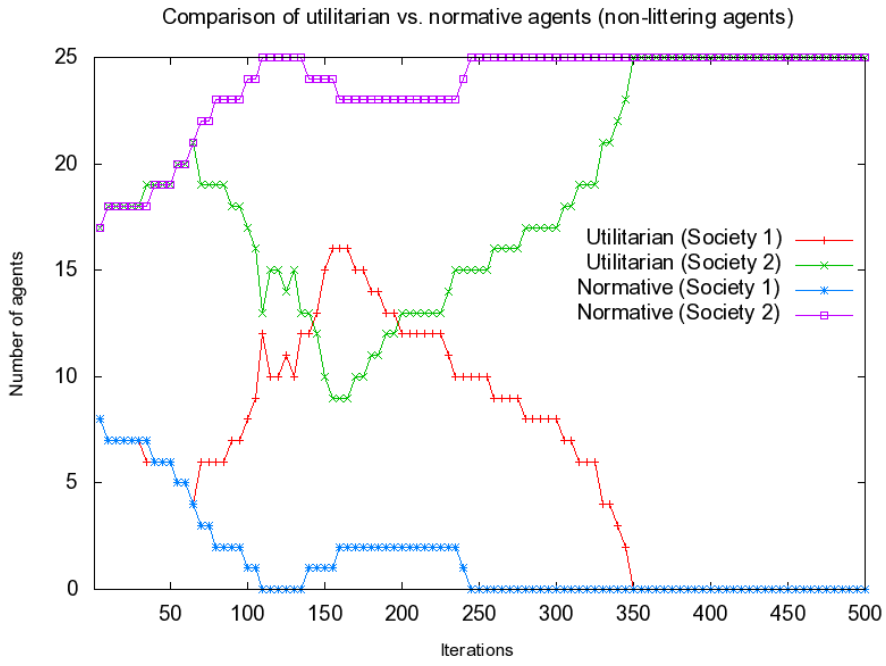

Figure 12: Comparison of utilitarion vs. normative-utilitarian strategies

7. Discussion

In this section we describe the main contributions of this paper, its limitations and the future work.

7.1. Contributions

The main contributions of this paper are the following:

1. The question of "how an agent comes to find out what the norms of society are" has not received much attention in the field of normative multi-agent systems. We have made progress in that regard by proposing a norm identification architecture from the perspective of a single agent (i.e. internal agent architecture for norm identification). Most researchers agree that there will be some form of sanction or reward once a norm is established (e.g. [10, 31]). Hence, the notion of a reaction (positive or negative action) has been considered to be a top level entity in our work. We have assumed that even when a norm is being created, the notion of sanction is important for norm identification.
2. The proposed architecture uses Candidate Norm Inference (CNI) algorithm to identify potential norms. The CNI algorithm makes use of a modified version of the association rule mining algorithm called WINEPI. To our knowledge this is the first work that makes use of a association rule mining approach for norm identification. An adaptive agent can infer norms by varying different parameters of the algorithm.
3. Through simulations we have shown how the architecture allows for detection (i.e. identification), communication (i.e. verification) and modification (i.e. dynamic change of norms) of norms. We have also demonstrated that the norm inference mechanism is beneficial for an agent as it learns about a norm faster than just using a utilitarian strategy.

We believe this architecture can be used in several settings. For example, the norm identification architecture can be used to infer norms in Massively Multi-player Online Role Playing Games (MMORPGs). Players involved in massively multi-player games perform actions in an environment to achieve a goal. They may play as individuals or in groups. When playing a cooperation game (e.g. players forming groups to slay a dragon), individual players may be able to observe prescriptions or proscriptions of actions that are allowed within the group. The normative architecture proposed in this paper can be used to identify norms that are being formed. Secondly, in virtual environments such as Second Life, norms can be inferred using this architecture. Thirdly, software agents engaging in e-commerce activities such as buying and selling goods, can infer norms using this architecture. Even though this paper focuses on prohibition norms, the same architecture can be used for identifying the violation of obligations. In this case, instead of looking for sequence of actions that could have caused a sanction, an agent can look for the absence of an event or a set of events that could have caused a sanction. In the e-commerce scenario it could be the absence of a *pay* action within certain period of time that could be a reason for a sanction.

7.2. Limitations and future work

The following are some of the limitations of this work.

Firstly, the issue of false positives and negatives has not been explicitly demonstrated in this work (e.g. a rogue agent punishing random actions). However, we believe this issue can be resolved at the norm verification stage (i.e. punishing agents can communicate the reason for punishment to the agents that ask for norm verification). Secondly, even though the cost of sanctions on the part of littering agents were considered, the cost of punishment on the punishing agents has not been considered. We assume that the punishers have other utility functions for punishing (e.g. a leader who wants to promote a smoother functioning of the society, or an altruistic agent who does not care about its diminishing utility). Thirdly, this work identifies co-existing norms (e.g. norm against eating coexisting with norm against littering). The architecture does not explain how an agent may be able to identify conflicting norms and how these conflicting norms can be handled by the agent[12]. Fourthly, we recognize that it can be difficult to always associate sanctions or rewards with the events that immediately precede them. For example, speeding might result in a fine that is sent to an agent after a couple of days. An observer might not be able to recognize this sanction. In this work, we have only considered those norms where the sanctions can be recognized by an observer and the events that caused the sanction occurred within a window of time (which can be varied by an agent) before the sanction. We intend to address these limitations in the future. In the future we will also investigate what an agent does with the norms once it has identified the norms (i.e. the norm applicability assessment component).

8. Conclusion

This paper addresses the question of how agents may infer a norm when joining a new society. To this end, this paper proposes the internal agent architecture for norm identification. An agent employing this architecture will make use of Candidate Norm Inference (CNI) algorithm to infer what the norms of the society are. The CNI algorithm identifies candidate norms based on an association rule mining approach. The architecture enables an agent to flexibly identify norms based on varying different parameters associated with the algorithm. Using experimental results it has been demonstrated that norm inference mechanism can be beneficial to an agent.

References

[1] F. López y López, Social Powers and Norms: Impact on Agent Behaviour, Ph.D. thesis, Department of Electronics and Computer Science, University of Southampton, United Kingdom (2003).

[12] We note that norm conflict resolution is being studied by researchers (e.g. [32]). Our interest is on identifying conflicting norms

[2] D. Gaertner, A. García-Camino, P. Noriega, J. A. Rodríguez-Aguilar, W. W. Vasconcelos, Distributed norm management in regulated multi-agent systems, ACM, Honolulu, Hawaii, 2007, pp. 624–631.

[3] M. Boman, Norms in artificial decision making, Artificial Intelligence and Law 7 (1) (1999) 17–35.

[4] H. Aldewereld, F. Dignum, A. García-Camino, P. Noriega, J. A. Rodríguez-Aguilar, C. Sierra, Operationalisation of norms for usage in electronic institutions, in: Proceedings of the fifth international joint conference on Autonomous Agents and MultiAgent Systems (AAMAS'06), ACM Press, New York, NY, USA, 2006, pp. 223–225.

[5] A. García-Camino, J. A. Rodríguez-Aguilar, C. Sierra, W. Vasconcelos, Norm-oriented programming of electronic institutions, in: Proceedings of the fifth international joint conference on Autonomous Agents and MultiAgent Systems, AAMAS, ACM Press, New York, NY, USA, 2006, pp. 670–672.

[6] J. Vázquez-Salceda, Thesis: The role of norms and electronic institutions in multi-agent systems applied to complex domains. the harmonia framework, AI Communications 16 (3) (2003) 209–212.

[7] G. Andrighetto, R. Conte, P. Turrini, M. Paolucci, Emergence in the loop: Simulating the two way dynamics of norm innovation, in: G. Boella, L. van der Torre, H. Verhagen (Eds.), Normative Multi-agent Systems, no. 07122 in Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.

[8] G. Andrighetto, M. Campenní, F. Cecconi, R. Conte, How agents find out norms: A simulation based model of norm innovation, in: Proceeding of the 3rd International Workshop on Normative Multiagent Systems (NorMAS), 2008, pp. 16–30.

[9] E. Ullmann-Margalit, The Emergence of Norms, Clarendon Press, 1977.

[10] J. Elster, Social norms and economic theory, The Journal of Economic Perspectives 3 (4) (1989) 99–117.

[11] G. Boella, L. Torre, H. Verhagen, Introduction to the special issue on normative multiagent systems, Autonomous Agents and Multi-Agent Systems 17 (1) (2008) 1–10.

[12] Y. Shoham, M. Tennenholtz, Emergent conventions in multi-agent systems: Initial experimental results and observations, in: Proceedings of third International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, San Mateo, CA, 1992, pp. 225–231.

[13] A. Walker, M. Wooldridge, Understanding the emergence of conventions in multi-agent systems, in: V. Lesser (Ed.), Proceedings of the First International Conference on Multi–Agent Systems, MIT Press, San Francisco, CA, 1995, pp. 384–389.

[14] J. M. Epstein, Learning to be thoughtless: Social norms and individual computation, Computational Economics 18 (1) (2001) 9–24.

[15] F. López y López, A. A. Márquez, An architecture for autonomous normative agents, in: Proceedings of the Fifth Mexican International Conference in Computer Science (ENC'04), IEEE Computer Society, Los Alamitos, CA, USA, 2004, pp. 96–103.

[16] G. Boella, L. van der Torre, An architecture of a normative system: counts-as conditionals, obligations and permissions, in: Proceedings of the fifth international joint conference on Autonomous Agents and MultiAgent Systems (AAMAS'06), ACM Press, New York, NY, USA, 2006, pp. 229–231.

[17] F. López y López, M. Luck, M. d'Inverno, Constraining autonomy through norms, in: Proceedings of The First International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS'02), 2002, pp. 674–681.
URL `citeseer.ist.psu.edu/lopez02constraining.html`

[18] R. Axelrod, An evolutionary approach to norms, The American Political Science Review 80 (4) (1986) 1095–1111.

[19] A. Staller, P. Petta, Introducing emotions into the computational study of social norms: A first evaluation, Journal of Artificial Societies and Social Simulation 4 (1).

[20] C. Scheve, D. Moldt, J. Fix, R. Luede, My agents love to conform: Norms and emotion in the micro-macro link, Computational and Mathematical Organization Theory 12 (2-3) (2006) 81–100.

[21] R. Conte, C. Castelfranchi, From conventions to prescriptions — towards an integrated view of norms ., Artif. Intell. Law 7 (4) (1999) 323–340.

[22] H. Verhagen, Simulation of the Learning of Norms, Social Science Computer Review 19 (3) (2001) 296–306.

[23] S. Sen, S. Airiau, Emergence of norms through social learning, in: Proceedings of Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07), AAAI Press, 2007, pp. 1507–1512.

[24] C. Castelfranchi, R. Conte, M. Paolucci, Normative reputation and the costs of compliance, Journal of Artificial Societies and Social Simulation 1 (3).

[25] B. T. R. Savarimuthu, S. Cranefield, A categorization of simulation works on norms, in: G. Boella, P. Noriega, G. Pigozzi, H. Verhagen (Eds.), Normative Multi-Agent Systems, no. 09121 in Dagstuhl Seminar Proceedings, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, Dagstuhl, Germany, 2009.
URL http://drops.dagstuhl.de/opus/volltexte/2009/1905

[26] M. Neumann, A classification of normative architectures, in: Proceedings of World Congress on Social Simulation, 2008.

[27] A. Ceglar, J. F. Roddick, Association mining, ACM Computing Surveys 38 (2) (2006) 5.

[28] H. Mannila, H. Toivonen, A. Inkeri Verkamo, Discovery of frequent episodes in event sequences, Data Mining and Knowledge Discovery 1 (3) (1997) 259–289.

[29] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: J. B. Bocca, M. Jarke, C. Zaniolo (Eds.), Proceedings of 20th International Conference on Very Large Data Bases (VLDB'94), Santiago de Chile, Chile, Morgan Kaufmann, 1994, pp. 487–499.

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

[31] J. Coleman, Foundations of Social Theory, Belknap Press, 1990.

[32] M. J. Kollingbaum, W. W. Vasconcelos, A. García-Camino, T. J. Norman, Managing conflict resolution in norm-regulated environments, in: Engineering Societies in the Agents World (ESAW'07), 2007, pp. 55–71.