JASSS

Bastin Tony Roy Savarimuthu, Stephen Cranefield, Maryam A. Purvis and Martin K. Purvis (2010)

Obligation Norm Identification in Agent Societies

Journal of Artificial Societies and Social Simulation 13 (4) 3
<http://jasss.soc.surrey.ac.uk/13/4/3.html>

Received: 18-Feb-2010 Accepted: 02-Jul-2010 Published: 31-Oct-2010

Abstract

Most works on norms have investigated how norms are regulated using institutional mechanisms. Very few works have focused on how an agent may infer the norms of a society without the norm being explicitly given to the agent. This paper describes a mechanism for identifying one type of norm, an obligation norm. The Obligation Norm Inference (ONI) algorithm described in this paper makes use of an association rule mining approach to identify obligation norms. Using agent based simulation of a virtual restaurant we demonstrate how an agent can identify the tipping norm. The experiments that we have conducted demonstrate that an agent in the system is able to add, remove and modify norms dynamically. An agent can also flexibly modify the parameters of the system based on whether it is successful in identifying a norm.

Keywords: Norms, Social Norms, Obligations, Norm Identification, Agent-Based Simulation, Simulation of Norms, Artificial Societies, Normative Multi-Agent Systems (NorMAS)

Introduction

1.1 Most works on norms in normative multi-agent systems have concentrated on how norms regulate behaviour (e.g. López y López 2003;Boman 1999;Vázquez-Salceda 2003). 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 (Boman 1999) or through an institution that prescribes what the norms of the society should be (Vázquez-Salceda 2003).

1.2 Only a few researchers have dealt with how an agent may infer what the norms of a newly joined society are (Andrighetto et al. 2007, Savarimuthu et al. 2009, 2010). Recognizing the norms of a society is beneficial to an agent. This process enables the agent to know what the normative expectation of a society is. 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 a changing environment. In open agent systems, instead of possessing predetermined notions of what the norms are, agents can infer and identify norms through observing patterns of interactions and their consequences. For example, a new agent joining a virtual environment such as Second Life (Rymaszewski et al. 2006) may have to infer norms when joining a society as each society may have different norms. It has been noted that having social norms centrally imposed by the land owners in Second Life is ineffective and there is a need for the establishment of community driven norms (Stoup, 2008). When a community of agents determines what the norms should be, the norms can evolve over time. So, a new agent joining the society should have the ability to recognize the changes to the norms. Identifying norms are also important because knowing the norms will help the agent not to lose utility as the agent can apply the norms when situations warrant it. Otherwise, the agent may be sanctioned for not following the norms.

1.3 This work aims to answer the question of how agents infer norms in a multi-agent society. To that end, the work described in this paper makes use of the norm identification architecture (Savarimuthu et al. 2010) we have proposed previously. The architecture is based on observation of interactions between agents and also recognizing signalling actions (sanctions and rewards). We have demonstrated how an agent can identify prohibition norms. In this work we present how an autonomous agent is able to identify obligation norms in a society using the Obligation Norm Inference (ONI) algorithm. When compared to identifying prohibition norms, identifying obligation norms is difficult because with a prohibition norm it is the occurrence of a particular event (or a sequence of events) that is the reason for a sanction to occur (e.g. dropping a litter is the reason for a sanction in a park). In obligation norms, it is the absence of an event that is the cause of a sanction (e.g. a waiter in a restaurant may sanction a customer for not tipping). It is difficult to detect the missing event (or a sequence of events) that is responsible for a sanction to occur. The ONI algorithm presented in this paper can be used to identify obligation norms in a society. Using a restaurant example, we show how an agent makes use of the norm identification framework to identify an obligation norm - the norm of tipping.

1.4 The paper is organized as follows. Section 2 provides a background on norms and normative multi-agent systems (NorMAS). Section 3 provides an overview of the norm identification framework. Section 4 describes the Obligation Norm Inference (ONI) algorithm and how the components of the framework are used in the context of a restaurant scenario. Section 5 describes the experiments that we have conducted and the results obtained. Section 6 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 7.

Background

2.1 In this section, we first provide a brief background on the role of norms in human societies. Second, we discuss the work on norms in the field of Normative Multi-agent Systems (NorMAS) and the relevance of our work in this area.

Norms in human societies

2.2 Due to multi-disciplinary interest in norms, several definitions for norms exist. Ullmann-Margalit (1977) 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 (1989) notes the following about social norms. "*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*".

2.3 Based on the definitions provided by various researchers, we note that the pragmatic aspects surrounding a social norm include the following:

- 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: Having obtained a norm, an agent may spread the norm to other agents in the society. Examples of norm spreading mechanisms include the notion of advice from powerful leaders, imitation and learning on the part of an agent.

2.4 Many social scientists have studied why norms are adhered to. Some of the reasons for norm adherence include a) fear of authority or power (Axelrod 1986;Jones and Sergot 1996), b) rational appeal of the norms (i.e. they promote self interest) (Akerlof 1976;Becker 1978) c) emotions such as shame, guilt and embarrassment that arise because of non-adherence (Elster 1989;Staller and Petta 2001;Scheve et al. 2006) and d) willingness to follow the crowd (Epstein 2001).

Normative multi-agent systems

2.5 The definition of normative multi-agent systems given by the researchers involved in the NorMAS 2007 workshop is as follows (Boella et al. 2008). 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 (Shoham and Tennenholtz 1992;Walker and Wooldridge 1995). Epstein has shown that norms reduce the amount of computation required to make a decision (Epstein 2001). 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.

2.6 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. López y López and Marquez (2004) have designed an architecture for normative BDI agents. Boella & van der Torre (2006) have proposed a distributed architecture for normative agents. For an overview and comparison of different architectures refer to Neumann's article (2010). Many researchers have used deontic logic to define and represent norms (García-Camino et al. 2006;Boella and van der Torre 2006). Several researchers have worked on mechanisms for norm compliance and enforcement (López y López et al. 2002 ;Aldewereld et al. 2006;Axelrod 1986). A recent development is the research on emotion-based mechanisms for norm enforcement (Staller and Petta 2001;Scheve et al. 2006). Conte and Castelfranchi (1999) 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.

2.7 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 (Verhagen 2001). In the bottom-up approach, the agents come up with a norm through learning mechanisms (Shoham and Tennenholtz 1992;Sen and Airiau 2007). Researchers have used sanctioning mechanisms (Axelrod 1986) and reputation mechanisms (Castelfranchi et al. 1998) for enforcing norms. An overview of different mechanisms used by researchers for the research on norms is provided by Savarimuthu and Cranefield (2009).

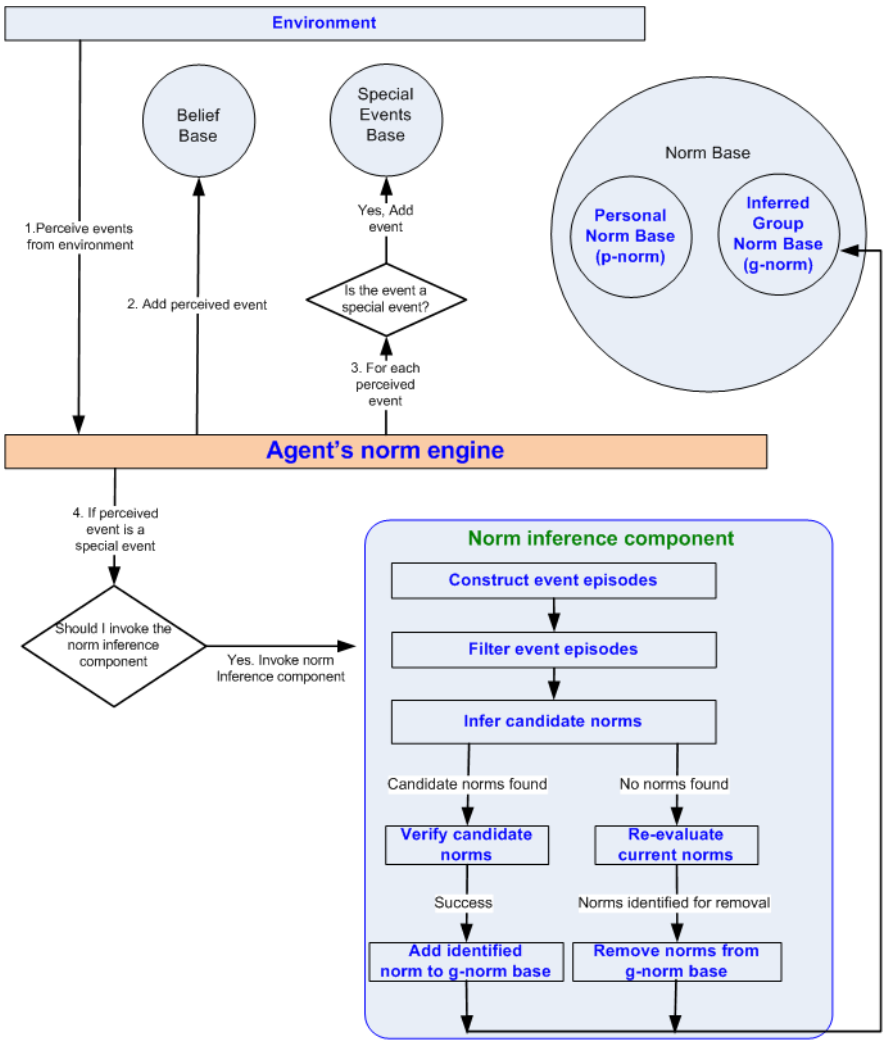
2.8 The work reported in this paper falls under the bottom-up approach to the study of norms. Many researchers in this approach have experimented with game-theoretical models for norm emergence (Axelrod 1986;Shoham and Tennenholtz 1992). Agents using these models learn to choose a strategy that maximizes utility. The agents in these works do not possess the notion of "normative expectation". Many research works assume that norms exist in the society and the focus is on how the norms can be regulated in an institutional setting such as electronic institutions (Arcos et al. 2005).

2.9 Very few have investigated how an agent comes to know the norms of the society (Andrighetto et al. 2007 ;Savarimuthu et al. 2009). Researchers involved in the EMIL project (Andrighetto et al. 2007) 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 work reported here differs from the work done in EMIL project in two 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 (Ceglar and Roddick 2006), a data mining technique, we propose an algorithm for norm inference, called the Obligation Norm Inference (ONI) algorithm, which can be adapted by an autonomous agent for flexible norm identification. To the best of our knowledge, we are the first to use a data mining approach for recognizing norms in an agent society.

Overview of the norm identification architecture

3.1 In this section we provide an overview of the norm identification framework (called the norm engine) proposed previously by Savarimuthu et al. (2010) for an agent to infer norms in the agent society in which it is situated. Social learning theory (Bandura 1977) suggests that new behaviour can be learnt through the observation of punishment and rewards. It has been noted that social monitoring (i.e. the observation of agent actions) (Conte and Dignum 2001) and social learning (Conte and Paolucci 2001) can be used to identify norms in the society. An agent employing the architecture makes use of social monitoring and learning to infer norms (Savarimuthu et al. 2010).

3.2 Figure 1 shows the internal agent architecture for norm identification. An agent employing this architecture follows a four-step process.



Environment
Belief Base
Special Events Base
Norm Base
Personal Norm Base (p-norm)
Inferred Group Norm Base (g-norm)
1.Perceive events from environment
2. Add perceived event
Yes, Add event
Is the event a special event?
3. For each perceived event
Agent's norm engine
4. If perceived event is a special event
Should I invoke the norm inference component
Yes. Invoke norm Inference component
Norm inference component
Construct event episodes
Filter event episodes
Infer candidate norms
Candidate norms found
No norms found
Verify candidate norms
Re-evaluate current norms
Success
Norms identified for removal
Add identified norm to g-norm base
Remove norms from g-norm base

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

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

3.4 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 a restaurant, a regular event is an event such as an agent ordering an item. "Special events" are signalling events that agents understand to be either encouraging or discouraging certain behaviour. For example when an agent orders a particular item but does not pay in accordance with a norm of the society, the agent can be sanctioned by the restaurant manager or the waiter. When the client is obliged to pay for the items ordered before leaving the restaurant, the waiter may shame the client by yelling or can even report this to authorities such as the police. Let us assume that the signal in this context is the occurrence of the shaming event which is a form of a sanction. The framework assumes that an agent has the ability to recognize signalling events based on its previous experience. Another example of a norm in this context is that a restaurant may have a social norm that the customers are expected to tip the waiter before departing the restaurant. A customer may be sanctioned by the waiter agent. The sanction here could be a *yell* or *shout* action.

3.5 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.

3.6 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* based on its past experience or preference. A *p-norm* is the personal value of an agent. For example an agent may consider that every agent should tip after having a meal in the restaurant. 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 group norm that an agent infers, based on its personal interactions as well as the interactions it observes in the society. An agent infers *g-norms* using the norm inference component of the framework.

3.7 The norm inference component is the main component of this architecture which employs sequence mining approach to identify norms (discussed in detail in Section 4). 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 the restaurant scenario, an agent observing a sanctioning event may invoke its norm inference component to find out what events that had happened in the past (or that had not happened in the past) may have triggered the occurrence of the special event. The invocation of the norm inference component may result in the identification of a *g-norm*, in which case it is added to the *g-norm* base.

3.8 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.

3.9 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, it removes the norm from the *g-norm* base. The operational details of the norm inference component are explained in Section 4.3. What an agent does with the norms once it has inferred the norms is out of the scope of this paper.

3.10 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.

Obligation norm identification

4.1 In this section we explain how obligation norms can be identified using the obligation norm inference (ONI) algorithm proposed here. First, we describe the domain in which an obligation norm is identified. Second, we describe how the events are stored by an agent. Third, we describe how the ONI algorithm can be used by an agent to infer norms.

Restaurant scenario

4.2 Let us assume that agents are situated in a restaurant in a virtual environment (e.g. Second Life). A new agent coming to a restaurant may not be aware of the protocol associated with ordering and paying for food items and the associated norms. For example, the protocol of a restaurant might be to first order and pay for the food before consuming the food while the protocol of another restaurant may be that the agent can consume the food and pay the bill at the end. A norm associated with the restaurants could be that the agent is expected to pay a tip after paying the bill or pay a tip along with the bill. This may vary from one culture to another. For example, in the USA it is expected to pay a tip while in New Zealand a tip is not expected. Depending upon one's location, failure to follow the norms may result in a sanction. In this paper we explain how an agent can identify the norms and the protocols in a particular context (i.e. the restaurant scenario).

Event storage components

4.3 Let us assume that an agent is situated in a restaurant where multiple agents come to the restaurant, consume food and move out of the restaurant. Let us also assume that a new agent is not aware of the norms or the protocols of the restaurant. 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 (e.g. eating). 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 (as events) in its belief base. An agent in the society can assume one or more of the following three roles: a participant (P) that is involved in a personal interaction, an observer (O) and a signaller (S). The participants are the waiter agent and the customer agent. The signaller is a waiter agent.

4.4 In the restaurant scenario, the agent is aware of the actions performed by an agent, which are the arrival of an agent (*arrive*), ordering an item (*order*), eating the ordered food (*eat*), paying for the ordered food (*pay*), tipping the waiter (*tip*) and departing the restaurant (*depart*). The agent also has the ability to recognize a signalling action such as *yell* or *shame* [1]. Signalling events can either be positive (rewards) or negative (sanctions). In this work we focus on the negative signals (i.e. sanctions)

4.5 Let us assume that a new agent (an observer) is situated in the restaurant. The observer records interactions that occur in the restaurant. Let us assume that a sanctioning event occurs. 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 because it only observes a sequence of actions and the agent does not know which of the events that happened in the past or the absence of which event(s) triggered the sanction). It will infer norms using the norm inference mechanism.

4.6 An event that is perceived by an agent consists of an event index, an observed action, and the agent(s) participating in that event. For example an agent observing an agent A arriving at the restaurant will represent this as *happens* (*1,arrive,A*). This implies the observer believes that the first event was generated by agent A which arrives in the restaurant.

4.7 A sample representation of events observed by an agent is given in Figure 2. An agent situated in an environment can sense these actions through observation or through action logs that may be available[2].

$$\left(\begin{array}{c} happens(1, arrive, A) \\ happens(2, arrive, B) \\ happens(3, order, A, W) \\ happens(4, eat, A) \\ happens(5, pay, A, W) \\ happens(6, depart, A) \\ happens(7, sanction, W, A) \\ happens(8, order, B, W) \end{array}\right)$$

Figure 2. Representation of events

4.8 An agent records these events in its belief base. Event 7 is a sanctioning event, where agent W sanctions agent A. The reason for the sanction is that agent A failed to tip agent W. For an observer it may not be possible to know the reason for this sanction unless it was specified a priori by the agent's designer. In open agent societies, the norms of the society may not be known to an agent ahead of time. Additionally, the norms may evolve over time. In order to infer a norm of the society the agent will make use of the norm inference mechanism proposed here.

4.9 The agents have a filtering mechanism, which identifies signalling events and stores them in the special events base. It should be noted that special events, such as *yell* and *shame*, are categorized by an agent as sanctioning events and they are stored in the special events base as a sanction event.

Norm inference component

4.10 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.

4.11 The norm inference component of an agent is made up of two sub-components. The first sub-component makes use of the Obligation Norm Inference (ONI) algorithm to generate candidate obligation norms. Candidate obligation norms are the norms that an agent identifies as potential candidates that may 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.

4.12 This sub-section is organized as follows. Firstly we explain the parameters of the ONI algorithm. Secondly we describe the internal details of the ONI algorithm using the restaurant example.

Definitions of parameters used in the algorithm

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

4.14 History Length (HL): An agent keeps a 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. An agent constructs an event episode (EE) based on events that are stored in its history. Construction of event episodes is described in the next sub-section.

4.15 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=8 is given in Figure 2.

4.16 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 event set can contain events such as yell (SES = { *yell*, *shame* }). 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 sanctions.

4.17 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 Figure 2 contains the following events[3], UES = { *arrive, order, eat, pay, tip, sanction, wait, depart* }.

4.18 Occurrence Probability (OP): The occurrence probability of an event E is given by the following formula.

```
OP(E) = Number of occurrences of E/Total number of events in ES
```

4.19 Window size (WS): When an agent wants to infer norms, it looks into its history for a certain number of recent events preceding a sanction. For example, if the WS is set to 3, an agent creates an event episode (EE) with the last three events that were exchanged between agents involved in the interaction (e.g. a pair of agents: the customer and the waiter) that precede a sanction. It should be noted that an EE is a subsequence[4] of ES.

4.20 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 in finding all sub-episodes[5] of an event episode that have at least a 50% chance of being a candidate norm. The algorithm uses the NIT on two occasions. As the values of NIT for each of the occasions can be varied by an agent, there are two variables used which are NITa, and NITb.

4.21 Norm Inference Frequency (NIF): An agent invokes the component periodically. The agent has a parameter called the norm inference frequency (NIF) that specifies the time interval between two invocations of the norm inference component.

Obligation Norm Inference (ONI) algorithm

4.22 In this section we describe the Obligation Norm Inference (ONI) algorithm.

Overview of the algorithm

4.23 There are four main steps involved in the Obligation Norm Inference algorithm (see Algorithm 1). First, event episodes of a certain length are extracted from event sequences that an agent observes. These event episodes are stored in the event episode list (EEL). Second, based on the events in the special event set (e.g. sanctioning events), the event episodes in EEL are separated into two lists. The first list, called the Special Event Episode List (SEEL) contains all event episodes that contain at least one sanctioning event. The second list, called the Normal Event Episode List (NEEL) contains all event episodes that do not contain sanctioning events. Third, using SEEL, all sub-episodes which have occurrence probabilities greater than or equal to NITa are extracted and stored in the Norm-Related Event Episode List (NREEL) based on a modified version of the WINEPI algorithm (Savarimuthu et al. 2010). The modification was to include the identification of sequences that are resultant products of permutation with repetition (discussed in Section 4.31). Fourth, for each event episode in NREEL, all supersequences are extracted from NEEL and stored in a temporary list called tempEEList. Based on the supersequences stored in tempEEList, the modified version of the WINEPI algorithm can identify all permutations of supersequences with occurrence probabilities greater than or equal to NITb. These are stored in the Candidate Obligation Norm List (CONL). These four steps are explained in detail in the following sub-sections. Note that a table containing all acronyms used in this paper and their expansions are given in the Appendix.

```
Algorithm 1: Obligation Norm Inference algorithm (main algorithm)
1 begin
2     Create event episodes list (EEL);
3     Create special event episodes list (SEEL) and normal event episodes
      list (NEEL);
4     Extract norm-related event episodes list (NREEL) from SEEL;
5     Create Candidate Obligation Norm List (CONL) from NEEL using
      NREEL ; /* Algorithm 2                                         */
6 end
```

Algorithm 1. Obligation Norm Inference algorithm (main algorithm)

Creating event episodes

4.24 An agent records other agents' actions in its belief base. We call the sequence of events that were recorded in the belief base event sequences (ES). Let us assume that there are four agents A, B, C and W in the restaurant as given in Figure 2. A and B are customers while W is the waiter agent. Let us assume that agent C is the observer. Agent A arrives first and shortly afterwards agent B arrives. Agent A orders food from agent W. Agent A eats and then pays. Agent A then departs. Agent A is sanctioned by W. Agent B orders food from agent W.

4.25 An agent has a certain history length (HL). An agent at any point of time stores the history of observed interactions for the length equal to HL. When the norm inference component is invoked, the agent extracts, from the recorded history (event sequences (ES)) the events that involved a pair of agents. We call the retrieved event sequence the event episode (EE). A sample event episode from the viewpoint of an observer (agent C) is given below. The set on the left hand side of the colon indicates that the agents involved in the interaction are A and W. To the right of the colon is the event episode. A hyphen separates one event from the next.

{A,W} : (happens(3, order, A,W) — happens(4, eat,A)— happens(5, pay, A,W)
— happens(6, depart,A) — happens(7, sanction,W,A))

4.26 Based on what an agent observes (e.g. the event sequence given in Figure 2), the observer may assume that something that agent A did in the past may have caused the sanction. It could also be the failure of agent A to perform (a) certain action(s) that might have caused a sanction. In this work we concentrate on the latter[6]. Agent C then extracts the sequence of events (the event episode) that involved A and W based on the event sequence stored in its history. To simplify the notation, only the first letter of each event will be mentioned from here on (e.g. p for pay) and also the agent names are omitted. As the sequence caters for temporal ordering of events, the event ids are omitted. Thus the event episode for interactions between agents A and W shown above will be represented as

{A,W} : o — e — p — d — s

4.27 Figure 3 shows a sample event episode list (EEL) that contains ten events involving a pair of agents that are observed by another agent where HL=6. Note that the Unique Event Set (UES) in this case includes events *a, o, e, p, t, d, w* and *s* which stand for *arrive, order, eat, pay, tip, depart, wait* and *sanction* respectively.

$$\begin{pmatrix} \{A,W\} : (e-p-d-s) \\ \{B,W\} : (e-p-t-d) \\ \{C,W\} : (a-o-e-p-t) \\ \{D,W\} : (a-o-e-p-d-s) \\ \{E,W\} : (p-d-s) \\ \{F,W\} : (o-e-p-d-s) \\ \{G,W\} : (a-o-e) \\ \{H,W\} : (a-o-e-p) \\ \{I,W\} : (a-d) \\ \{J,W\} : (a-o-e-p-t-d) \end{pmatrix}$$

Figure 3. Event episode list (EEL)

Creating special and normal event episode lists

4.28 Note that some event episodes in EEL have sanctions as one of the events. The agent identifies the sanction events from the special events set (SES). Using EEL, an agent creates two lists for further processing, one with event episodes that contain a sanctioning event and the other containing event episodes without sanctions. The list that contains event episodes with sanctioning events is called the special event episode list (SEEL). The other list is called the normal event episode list (NEEL).

4.29 The SEEL obtained from our example EEL is given in the left in Figure 4. NEEL has the remaining episodes that do not contain a sanctioning action (shown in the right of Figure 4).

$$\begin{pmatrix} e-p-d-s \\ a-o-e-p-d-s \\ o-e-p-d-s \\ p-d-s \end{pmatrix}, \begin{pmatrix} e-p-t-d \\ a-o-e-p-t \\ a-o-e \\ a-o-e-p \\ a-d \\ a-o-e-p-t-d \end{pmatrix}$$

Figure 4. SEEL (left) and NEEL (right)

Generating the norm-related event list (NREEL)

4.30 From the SEEL, an agent can identify events that have the potential to be associated with sanctions. For example, from the SEEL shown in the left of Figure 4, the agent may infer that the sub-episodes *p-d, p*, or *d* could be the reason for a sanction as they occur in all the event episodes in SEEL. In the case of prohibition norms the events that precede a sanction can be potentially linked to sanction due to causality. In the case of obligation norms, it is the absence of an event or a sequence of events that might be the cause of the sanction. In both these types of norms, the agent has to identify the sequences of events that occur frequently before the occurrence of a sanctioning action. In the case of a prohibition norm, the frequency of occurrence may correlate with norm identification. In the case of an obligation norm, the agent first has to find the frequently occurring sequence(s), which are then stored in the norm-related event list (NREEL). Let us refer to an event episode in NREEL as α. Second, an agent has to identify all the supersequences of α in NEEL with an occurrence probability greater than or equal to NITa, which are added to the candidate obligation norm list (CONL). The construction of NREEL is discussed in this sub-section and the construction of CONL is discussed in the next sub-section.

4.31 In order to identify these norm-related events the agent uses the Candidate Norm Identification (CNI) algorithm (Savarimuthu et al. 2010) a modified version of the WINEPI algorithm (Mannila et al. 1997) which is based on association rule mining. Association rule mining (Ceglar and Roddick 2006) is one of the well known fields of data mining where relationships 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. Some well known algorithms in the data mining field can be used for mining frequently occurring episodes (i.e. mining association rules) (Agrawal and Srikant 1994;Mannila et al. 1997). A limitation of the well-known Apriori algorithm (Agrawal and Srikant 1994) is that it considers combinations of events but not permutations (e.g. it does not distinguish between event sequences *p-d* and *d-p*). WINEPI (Mannila et al., 1997) addresses this issue, but it lacks support for identifying sequences that are resultants of permutations with repetition. For example, from sub-episodes of length one, e.g. *p* and *d*, the algorithm can generate sub-episodes of length two which are *p-d* and *d-p*, but not *p-p* and *d-d*. Permutations with repetition are important because there could be a norm which sanctions an agent from performing the same action twice. The modification, reported previously (Savarimuthu et al. 2010), can identify candidate norms that are obtained by considering "permutations with repetition" when constructing sub-episodes.

4.32 Based on the SEEL, an agent can generate the NREEL. Figure 5 shows the SEEL on the left of the arrow and the NREEL generated from the SEEL on the right of the arrow when NITa is set to 0. The occurrence probability of an event episode in NREEL is given in square brackets. When NITa is set to 0, all possible subsequences of event episodes in the SEEL are generated. When NITa is set to 100 the algorithm identifies the following norm-related event episode list { *p-d, p, d* }. An agent, being an autonomous entity, can vary the NITa parameter to identify the norm-related events. Note that if an event episode is frequent then all its subsequences are also frequent. For example if *p-d* appears 100% of the time (i.e. the occurrence probability is 1), all its subsequences also appear 100% of the time.

$$\begin{pmatrix} (e-p-d-s) \\ (a-o-e-p-d-s) \\ (o-e-p-d-s) \\ (p-d-s) \end{pmatrix} \rightarrow \begin{pmatrix} (p-d)[1] \\ (p)[1] \\ (d)[1] \\ (e-p-d)[.75] \\ (e-p)[.75] \\ (e)[.75] \\ (o-e-p-d)[.5] \\ (o-e-p)[.5] \\ (o-e)[.5] \\ (o)[.5] \end{pmatrix}$$

Figure 5. SEEL (left) and NREEL (right)

Generating the candidate obligation norm list (CONL)

4.33 The pseudocode for generating CONL is given in Algorithm 2. In order to identify the obligation norms, the agent has to identify those supersequences in NEEL that contain the event episodes in NREEL. These supersequences are stored in a list (tempEEList in this case).

Algorithm 2: Pseudocode to create the candidate obligation norm list (CONL)

```
Input: Norm-Related Event Episode List (NREEL), Normal Event
       Episode List (NEEL), Norm Identification Threshold (NIT)
Output: Candidate Obligation Norm List (CONL)
1  CONL = $\emptyset$;
2  for each event episode $NREE \in NREEL$ do
3      tempEEList= $\emptyset$;
4      foreach event episode $EE \in NEEL$ do
5          if EE is a supersequence of NREE then
6              Add EE to tempEEList;
7          end
8      end
9      Use modified WINEPI algorithm to extract all candidate obligation
       norms (Input: tempEEList, Unique Event Set (UES), Window
       Size(WS), Norm Inference Threshold ($NIT_b$), Output: Candidate
       norms);
10     Add candidate obligation norms to CONL;
11 end
12 return CONL;
```

Algorithm 2. Pseudocode to create the candidate obligation norm list (CONL)

Based on the supersequences stored in tempEEList, the Candidate Norm Inference (CNI) algorithm (Savarimuthu et al. 2010) can identify all permutations of supersequences whose occurrence probabilities are greater than or equal to NITb. Such supersequences are stored in the candidate obligation norm list (CONL). For example, let us suppose that the event epsiode p-d is the only event episode stored in the NREEL. Figure 6 shows the NEEL on the left of the arrow and the tempEEList that is generated from the NEEL on the right. Note that the NEEL on the left contains six event episodes but tempEEList contains two event episodes that contain p-d out of six. These two event episodes are supersequences of *p-d*.

$$\begin{pmatrix} (e-p-t-d) \\ (a-o-e-p-t) \\ (a-o-e) \\ (a-o-e-p) \\ (a-o-d) \\ (a-o-e-p-t-d) \end{pmatrix} \rightarrow \begin{pmatrix} (e-p-t-d) \\ (a-o-e-p-t-d) \end{pmatrix}$$

Figure 6. NEEL (left) and tempEEList (right)

4.34 From tempEEList the CONL can be generated. The left hand side of Figure 7 shows the tempEEList. The right hand side of Figure 7 contains all permutations of supersequences of *p-d* that can be obtained from tempEEList and their occurrence probabilities in tempEEList (in square brackets). Assuming that NIT $_b$ is set to 100, the supersequences that will be identified as members of CONL are *p-t-d* and *e-p-t-d*. Both these supersequences have an occurrence probability of 1. As the occurrence probabilities of *o-e-p-t-d* and *a-o-e-p-t-d* are less than NITb, these are not included in the CONL. Note that the modified WINEPI algorithm is used twice, the first time to obtain the NREEL from the SEEL (not shown here) and the second time for obtaining the CONL from the NEEL using the NREEL (line 9 of Algorithm 2). For every event episode in the NREEL, a new CONL is generated. Having compiled a set containing candidate obligation norms, the agent passes this information to the norm verification component to identify norms. This process is iterated until there are no elements in NREEL. The norm verification process is explained in the next sub-section.

$$\begin{pmatrix} (e-p-t-d) \\ (a-o-e-p-t-d) \end{pmatrix} \rightarrow \begin{pmatrix} (p-t-d)[1] \\ (e-p-t-d)[1] \\ (o-e-p-t-d)[0.5] \\ (a-o-e-p-t-d)[0.5] \end{pmatrix}$$

Figure 7. *tempEEList* (left) and permutations of supersequences containing p-d in*tempEEList* (right)

Norm verification

4.35 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 in every 10 iterations). An agent A can ask another agent B, by choosing the first candidate norm (say p-t-d) for which it has the highest occurrence probability and asks B if it knows whether the obligation norm $O_{A,W}$ (t|p) is a norm of the society (i.e. an agent is obliged to tip after paying for the food ordered). If the response is affirmative, A stores this norm in its set of identified norms. If not, A moves on to the next candidate norm in its list. In the case of the running example, the second candidate norm *e-p-t-d* is chosen to be communicated to the other agent. It asks another agent (e.g. the agent that is the closest) whether it thinks that the given candidate norm is a norm of the society. If it responds positively, the agent infers $O_{A,W}$ (t|(e — p)) to be a norm. If the response is negative, this norm is stored in the bottom of the candidate norm list. This process continues until a norm is found or no norm is found from the event episodes in the candidate norm list. Even in the case of successfully identifying a candidate norm, the agent continues the process to identify any co-existing norms.

$$\begin{pmatrix} p - t - d \\ e - p - t - d \end{pmatrix} \rightarrow \begin{pmatrix} t \end{pmatrix}$$

Figure 8. Candidate norms (left) and identified norm (right)

4.36 Note that an agent will have two sets of norms: candidate norms and identified norms. Figure 8 shows the two sets of norms: the candidate norms on the left of the arrow and the identified norm on the right. 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).

Discussion on norm verification

4.37 The reason for having a norm verification procedure is that the data mining approach isn't sufficient to solve the norm identification problem to the fullest extent. For example let us assume that *p-t*, *p* and *t* are identified as candidate norms (which occur 100% of the time). It can be that only one of these is a norm in the society. While the data mining approach has narrowed down the search space to 3 possible choices, it does not suggest which one of these three is the actual norm (because all the three are equally significant from a data mining perspective). To solve this problem, we have used a social mechanism where agents ask other agents for norm verification. There are two other reasons for employing another agent as the norm verifier.

4.38 First, an agent entering a society may not be interested to find out all the norms of a society (an agent might give a long list of norms followed in the society). It may be interested to find the norms in a particular context. An agent has to first infer what the context is (by observing the interactions) and then it can ask another agent in the neighborhood if its inference of a norm is valid (e.g. *Am I obliged to tip in this society?*). In our view this is more effective (in terms of computation and memory required) than asking another agent what the norms in the restaurant are, as there could be a long list of norms that apply and most of those may not be of interest to an agent. As the agent has gathered some evidence with regards to what the norm could be, the agent has not only identified the context it is in but also can be confident in asking for norm referral (i.e. because it can precisely formulate its query for norm identification and also can be confident in its query which based on inference).

4.39 Second, an agent may not completely trust other agents in an open society. When an agent asks another agent without norm inference, the other agent could potentially lie about a norm. So, an agent may want to make sure that it identifies candidate norms before it asks for norm verification. This process helps an agent from being cheated by the referrer agent if it were to ask what the norm is as it knows that one of the candidate norms could potentially be a norm. Note that this doesn't solve the lying problem as the referrer agent can lie when an agent asks if something is a norm in the society. At the least, the mechanism allows the agent to have a set of candidate norms. The problem of lying can be addressed in two ways. First, an agent could ask for norm verification from many agents in its neighbourhood. Second, an agent can verify whether its candidate norms hold by undertaking actions that it observes to be sanctioned (e.g. by violating the tipping norm). Based on the outcome of tests the agent carries out it can infer what the norms could be. This is a meta-level norm testing mechanism of an agent. These mechanisms can be explored in the future.

Experimental results

5.1 In this section we first describe the experimental set-up in paragraphs 5.2 and 5.3. In the rest of the sub-sections we describe the experiments that were conducted and also discuss the results obtained.

Experimental set-up

5.2 We model agents in our virtual society in a two-dimensional space. This virtual restaurant environment is shown in Figure 9. The agents can enter the restaurant and occupy one of the chairs at a table. Each table has six chairs. Each agent has a visibility threshold. The visibility threshold of the agent is governed by a Moore neighbourhood (Weisstein, 2010) of radius *r*. An agent can observe actions of agents and the interactions that happen between two agents within its visibility threshold. There are three types of agents in the simulation. They are non-tipping customers (NTC), tipping customers (TC) and waiters (W). There are eight possible types of actions defined in the simulation system: arrive, order, eat, pay, tip, depart, wait and sanction. The NTC agents can *arrive, order, eat, pay* and *depart*. The TC agents can *arrive, order, eat, pay, tip* and *depart* while the waiter agents can *wait* on the customers and *sanction*[7]. The agents in this environment move from left to right. An agent chooses a seat and occupies it. It can then order food, eat and then pay for the food. The agent may tip the waiter. The agent may be sanctioned for not tipping. The agent can then depart the restaurant. The agents that are at the edge of the two dimensional space can again re-appear in the opposite side (i.e. a toroidal grid is implemented). The agents are represented as circles using different colours. The NTCs are red, the TCs are green and the Ws are blue. The id of an agent and the action it currently performs appear above the circle. At any time step an agent can perform one action. When an agent does the same action over several steps it is recorded as one action. For example, if the agent eats for 10 iterations, the eating action is counted as one action[8]. The same holds for the arrival and the departure of an agent. All the agents make use of the norm inference component to infer norms. The blue squares that appear within the circles represent the identification of a norm.

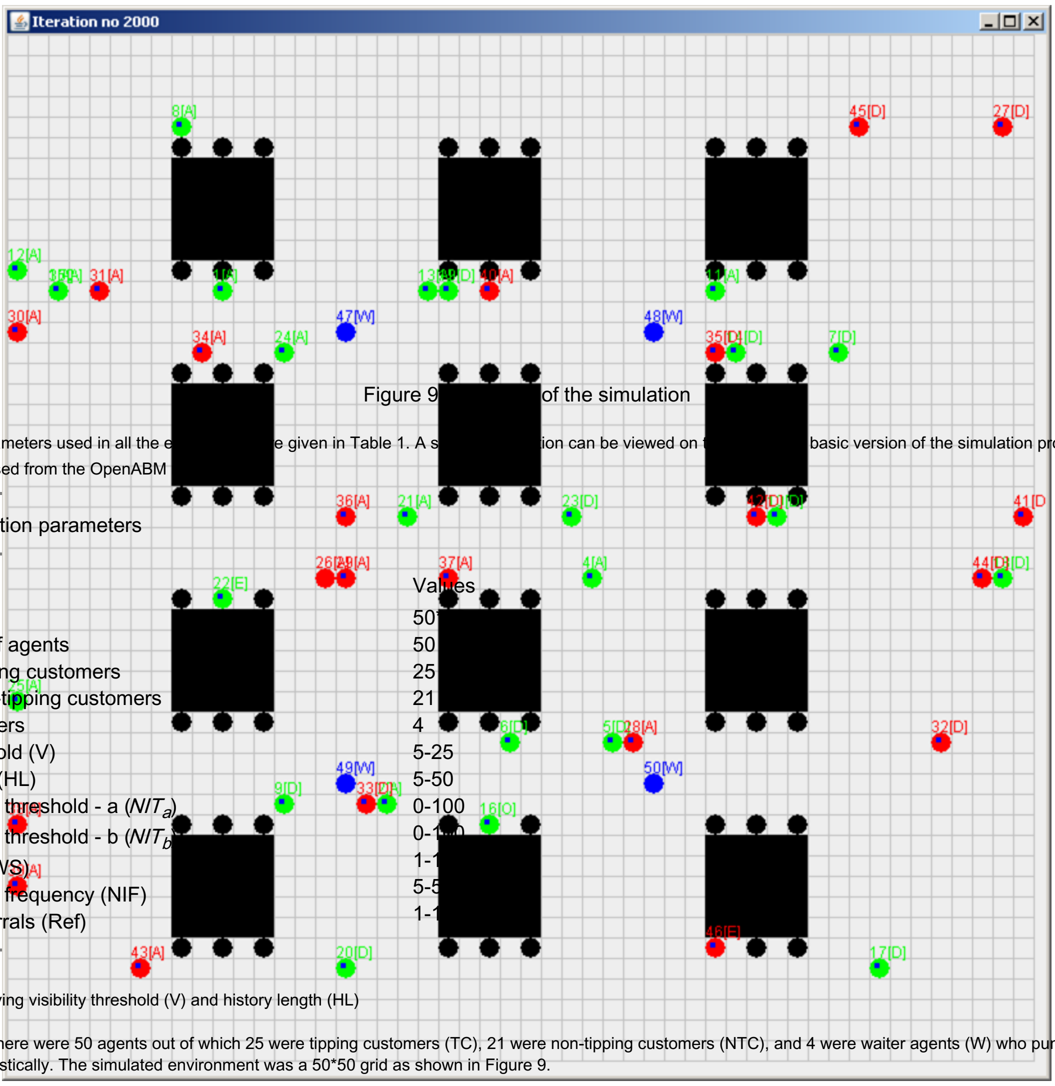


Figure 9 of the simulation

5.3 The simulation parameters used in all the e e given in Table 1. A s ion can be viewed on t basic version of the simulation program written in Java can be accessed from the OpenABM

Table 1: Simulation parameters

Parameters	Values
Grid size	50
Total number of agents	50
Number of tipping customers	25
Number of non-tipping customers	21
Number of waiters	4
Visibility threshold (V)	5-25
History Length (HL)	5-50
Norm inference threshold - a (NIT_a)	0-100
Norm inference threshold - b (NIT_b)	0-1 0
Window Size (WS)	1-1
Norm inference frequency (NIF)	5-5
Number of referrals (Ref)	1-1

Experiment 1 - Varying visibility threshold (V) and history length (HL)

5.4 In this experiment there were 50 agents out of which 25 were tipping customers (TC), 21 were non-tipping customers (NTC), and 4 were waiter agents (W) who punished non-tipping customers probabilistically. The simulated environment was a 50*50 grid as shown in Figure 9.

5.5 An agent has a visibility threshold which dictates how many cells an agent can see. A visibility threshold of 5 would mean that the agent can see all agents which are at the maximum five cells away from it on all sides. The agent also has certain amount of history regarding the actions performed by other agents. When the history length is five, an agent stores the actions of all the agents within its vicinity in the last 5 iterations.

5.6 Figure 10 shows the rate at which a society identifies the obligation norm of tipping when the visibility and the history lengths of agents are varied. In all the graphs in Figure 10, the x-axis shows the iteration number and the y-axis shows the number of agents with the tipping norm.

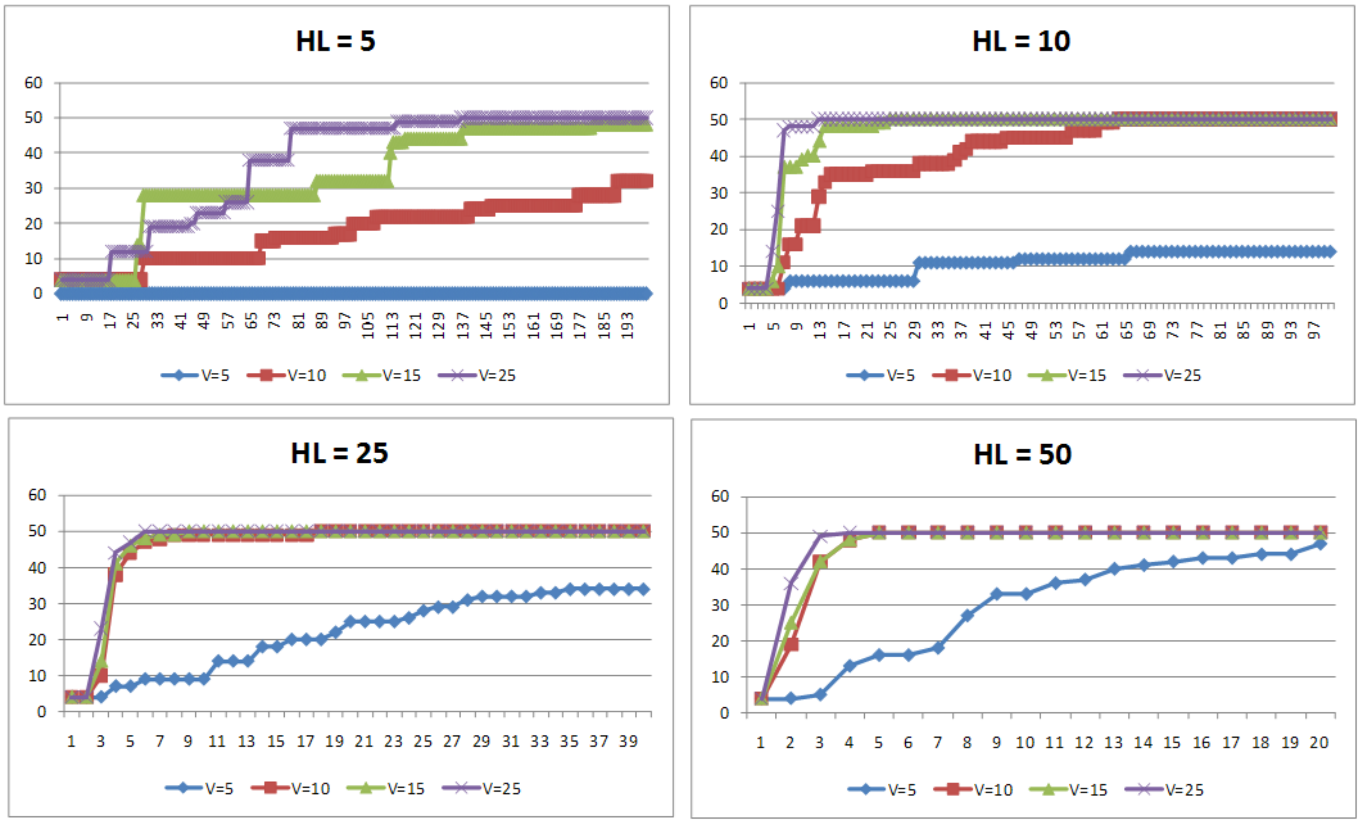


Figure 10. Varying the visibility threshold and history length of agents

5.7 By keeping the history length constant and all the other parameters constant we varied the visibility threshold for the agents. The top-left graph of Figure 10 shows the result of varying the visibility threshold. It can be noted that as the visibility of the agents increased, the agents identified the norms faster. This is because the agents were able to collect more evidence[11]. This can be observed in all the graphs in Figure 10.

5.8 When the history length of an agent was increased the agents in the society inferred the norms faster. When we compare the results shown in the top-right graph of Figure 10 with the results shown in the top-left graph of Figure 10, it can be observed that as the history length increases the agents infer the norms faster. This can be observed as we move from the top-left graph to the bottom-right graph in Figure 10. When the history length was small, the event episode list may not contain all the information to infer a norm. But when the history length is increased, an agent will have better evidence to infer the norms.

Experiment 2 - Varying referral levels

5.9 Once an agent has identified a candidate norm, it asks other agents for norm identification within its visibility threshold. An agent can vary this parameter. It was noted that as the number of agents from whom an agent asked for referral increased (see Figure 11), the norm identification rate of the agents in the society increased[12]. The norm inference frequency in this experiment was once every 50 iterations. Other parameters of this experiment were: NITa =100, NITb = 80, V=25, HL=50 and WS=3.

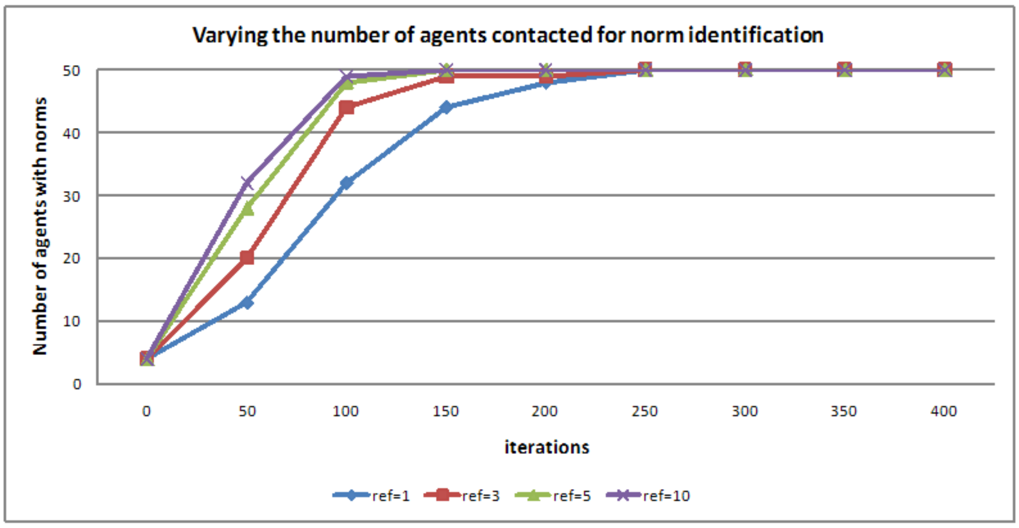


Figure 11. Varying the number of agents contacted for norm identification

Experiment 3 - Varying Norm Inference Thresholds (NIT)

5.10 We have studied the effect of changing the NIT thresholds (a and b) on the size of NREEL and CONL that are generated by an agent. Figure 12 shows the size of the NREEL when NIT_a is varied. When NIT_a is set low, the size of NREEL generated by an agent is large. This means that an agent incurs a large amount of computation cost to generate NREEL. When an agent sets NIT_a high, the size of NREEL is small. An agent, being an adaptive entity, can vary this parameter depending upon its success in identifying a norm.

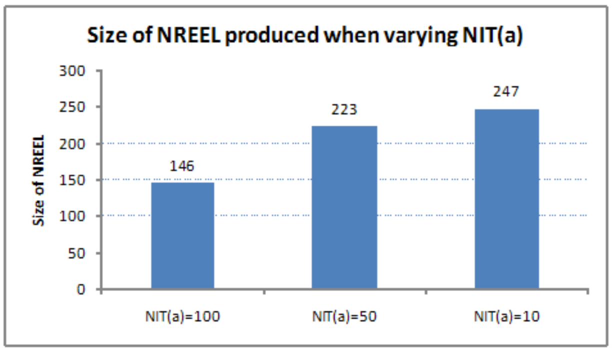


Figure 12. Size of NREEL when varying NIT_a

5.11 We also conducted experiments to show the impact of varying both NIT $_a$ and NIT$_b$ on the number of candidate norms generated (see Figure 13). When NIT $_a$ was set to 100 and NIT $_b$ was set to 100, a small set of candidate norms was generated (size = 9). When NIT$_a$ was set to 100 and NIT $_b$ was set to 50, 35 different candidate norms were generated. When NIT $_a$ was set to 50 and NIT$_b$ was set to 100, 37 candidate norms were generated. When NIT $_a$ was set to 50 and NIT $_b$ was set to 50, 57 candidate norms were generated. If an agent sets the NIT value high, the number of candidate norms that is generated is low. The number of candidate norms that are generated has an impact on the norm verification stage. The less the number of candidate norms, less the amount of time taken for norm verification. An agent can change these two parameters to adapt to the current situation. For example, if an agent sets the NIT values to be high and it does not find a norm it can decrease the value for the next norm inference instance.

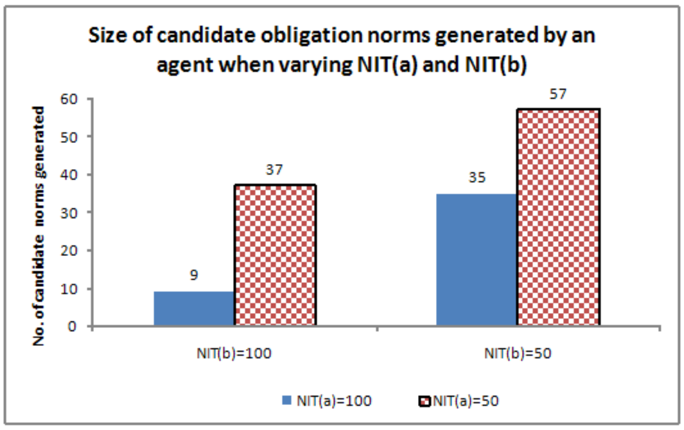


Figure 13. Number of candidate norms generated when varying NIT$_a$ and NIT$_b$

Experiment 4 - Varying the Window Size (WS)

5.12 The objectives of this experiment are two-fold. They are to show that

- As WS increases, the accuracy of norm identification increases (i.e. the number of false positives decreases and the number of candidate norms generated decreases)
- As WS is varied, different normative protocols associated with the norms can be identified.

Accuracy of norm identification

5.13 By keeping other parameters constant we varied the Window Size (WS) of norm identification. The results of varying an agent's WS for 20 norm inference instances [13] is given in Figure 14. The success rate in identifying three different categories of norms are also given in Figure 14. These three categories are 1) the only candidate norm identified is the tipping norm 2) tipping is identified as one of the candidate norms and 3) the tipping norm is not found or no norm is found). When WS is set to 1, the agent identified the tipping norm to be the only candidate norm 5% of the time. 65% of the time, it identified the tipping norm as one of the candidate norms. In other words, the agent also had identified other candidate norms. These other candidate norms are false positives which are pruned during the norm verification stage. Remaining 30% of the time, the agent either did not identify any norms or did not identify tipping as one of the norms.

5.14 As WS is increased, the accuracy of norm identification increases (i.e. the success of the agent in identifying tipping as the only norm increases). When WS=2, the agent identifies tipping to be the only norm 30% of the time. When WS=3, the agent identifies it 55% of the time and when WS is set to 5 it identifies it 85% of the time. It should be noted that as WS increases, the false positives decrease.

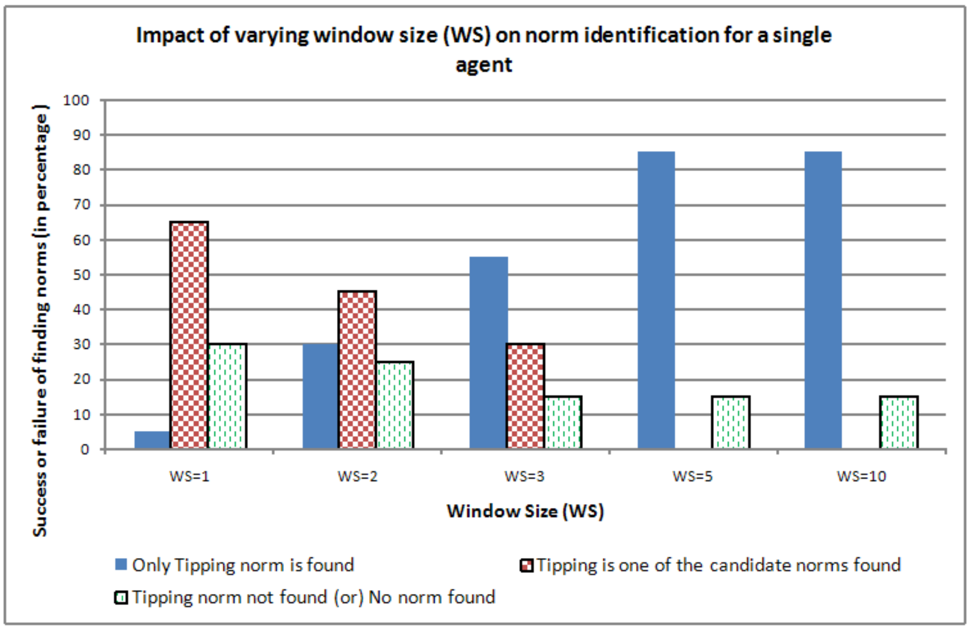


Figure 14. Impact of varying the Window Size (WS) on norm identification

5.15 When WS increases, the agent's load in norm verification decreases (shown in Figure 15). When WS=1, about four candidate norms were generated in each norm inference cycle. When WS=2, on average, more than two candidate norms were generated in each cycle and when WS is set to three, fewer than 2 norms were generated in each iteration. When WS was set to 5, one norm was generated in each iteration. When the number of candidate norms generated is less, the amount of norm verification that needs to be done is less which results in a reduction of communication between the agents.

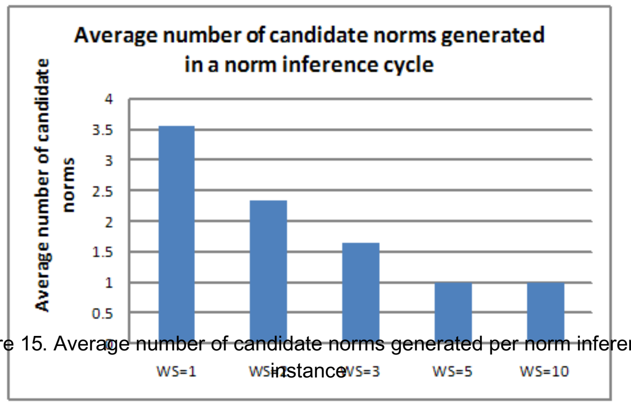


Figure 15. Average number of candidate norms generated per norm inference instance

5.16 There is no further improvement in the success of tipping norm identification when moving from WS=5 to WS=10 because the domain model supports a maximum of five different events that occur before a sanction (i.e. *a-o-e-p-d*). If the domain model is changed, the WS will need to be changed accordingly.

Identifying normative protocols

5.17 We define normative protocols to be the order of the occurrence of events (protocols) associated with a norm. For example, the protocol *a-o-e-p-t-d* defines the sequence that normally an agent arrives, occupies a seat and orders food, eats, pays, tips and then departs. The focus of this experiment is to demonstrate that an agent, by changing the WS, can have a partial or complete view of what the normative protocol might be. For example, when WS is set to 1, the agent identifies only one event that precedes the sanctioning event. Hence, the size of the normative protocol identified will be two[14]. If WS is set to five, the size of the normative protocol that can be identified can vary from two to six [15]. Figure 16 shows the normative protocols generated by an agent over 20 norm inference instances.

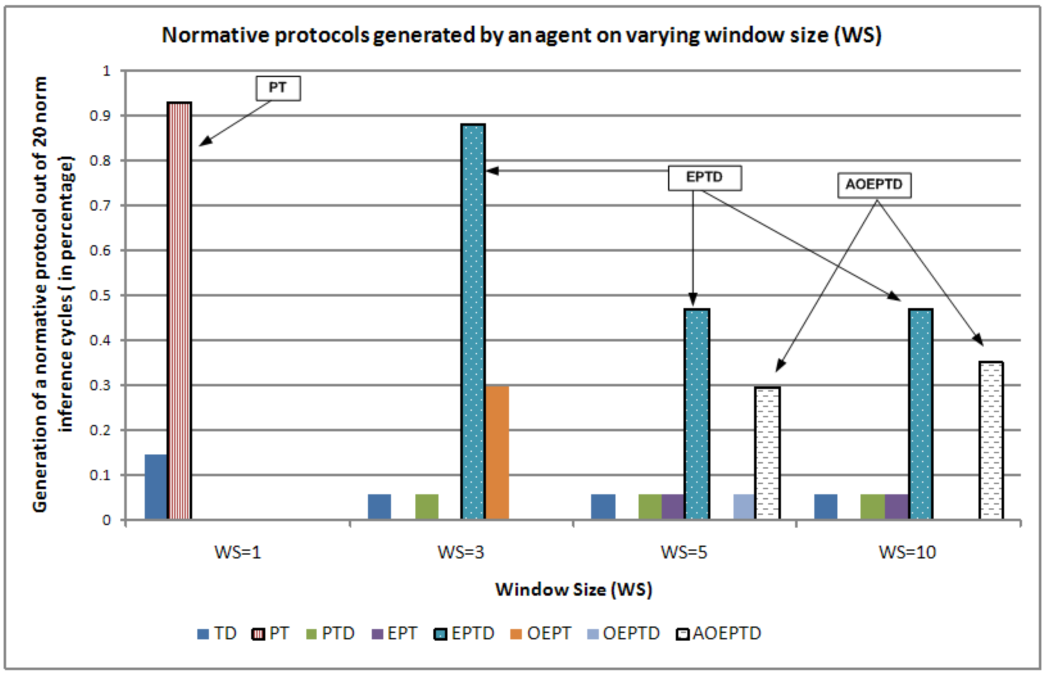


Figure 16. Normative protocols generated by an agent for varying window sizes

5.18 From Figure 16, it can be observed that when WS is one, the agent identified PT [16] as the protocol, about 90% of the time. When WS was set to three, it identified, EPTD to be the normative protocol about 88% of the time. When WS was set to five and ten, EPTD and AOEPTD were the top two normative protocols that were identified. It should be noted that as WS increases, the size of the normative protocol identified increases. Also, the normative protocols identified for lower values of WS are subsets of the normative protocols identified using higher values of WS.

5.19 This experiment shows that an agent not only infers norms but also the associated normative protocol. An agent's view of the normative protocol depends on the WS value of that agent. Note that, given a protocol, an agent can easily identify the norm associated with that protocol. For example, assuming that an agent knows the protocol in a restaurant is *a-o-e-p-d* (e.g. if this is given to the agent by the designer), an agent can easily identify *a-o-e-p-t-d* as the normative protocol based on observations of events. In our case, the normative protocol is inferred by the agent, without the protocol being given to the agent explicitly.

5.20 The norm and the protocol are inferred through the norm identification mechanism. If the normative protocol were to be *a-o-p-e-t-d* in a restaurant (i.e. pay before eating and then tip), our approach will be able to identify the norm and the normative protocol.

Experiment 5 - Adaptability of an agent in identifying norms

5.21 An agent in our system can flexibly modify the history length (HL) based on whether it is successful in identifying a norm. If an agent does not infer norms when HL=25, it can increase the HL. If it has identified if the norm holds at a particular HL, the agent will check after a certain number of iterations whether the norm still holds. If it holds, it will try to decrease the HL and check whether the norm can be identified. If it can be identified the agent will decrease its HL further. The objective for reducing the HL is to reduce the amount of computation required to find a norm. The agent will be better off in terms of the computation required if it can find the same norm when it lowers the amount of history it has to store.

5.22 The top line in Figure 17 shows the adaptive history length of an agent when it tries to identify a norm. The bottom line shows whether the agent has found a norm (a value of 5) or not (a value of 0). An agent initially starts with an HL of 25. When it does not find the norm when HL=25, it increases its HL by five. This increases to a maximum value of 50. Once a norm is found, the agent tries to check whether the same norm can be found for a lower value of HL. It can be inferred from Figure 17 that in iteration 75, the agent found the norm when the HL was 40. When it tried to reduce HL to 35, the norm was not found. The agent then increased the HL to 50 by incrementing it by 5 in the next few norm inference instances. The agent was able to find the norm again in iteration 250. It then decreased the HL to 45 and it found the norm again. When HL was set to 40 it did not find the norm, hence the agent increased HL to 45. This graph shows that an agent is adaptive in terms of the history length it stores. Dynamic adjustment of history length will be beneficial to the agent when norms are changing.

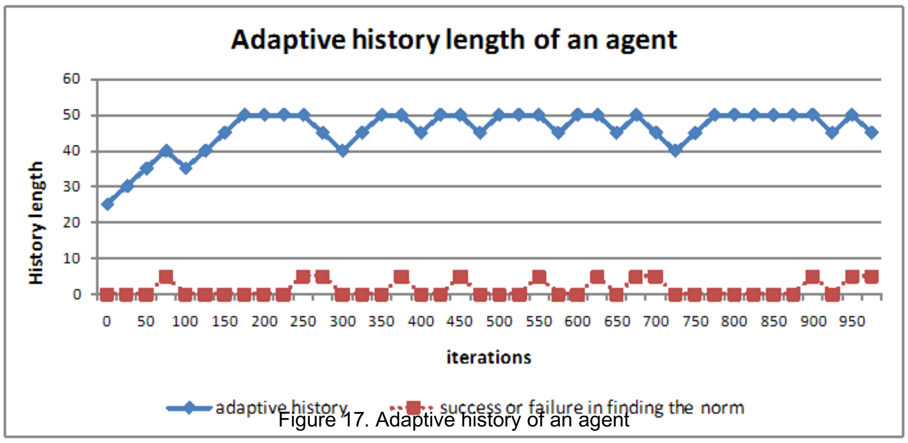
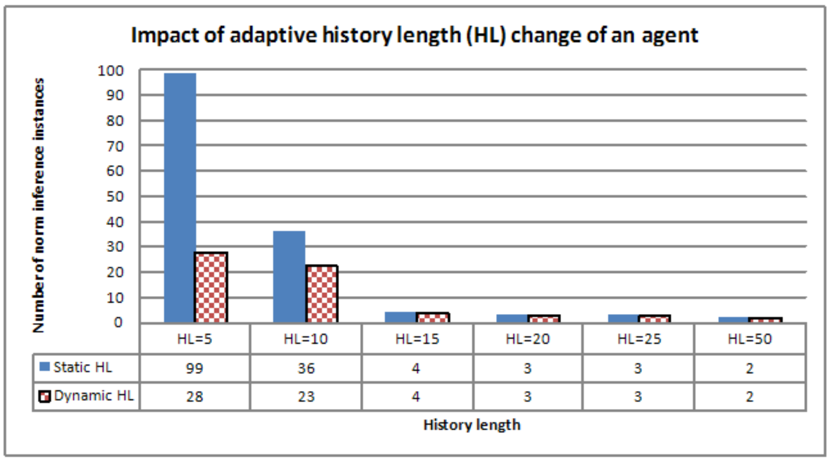


Figure 17. Adaptive history of an agent

5.23 We have also experimented with varying the history lengths of an agent with and without the ability of having an adaptive history length (i.e. static HL vs. dynamic HL). When HL is static the agent has a constant HL throughout the simulation. When HL is dynamic, the agent can change its HL based on whether it has identified a norm. It can be seen from Figure 18 that when dynamic HL is used, an agent is able to infer the norms faster. Note that when HL was set to five (static HL=5), the agent found the norm only after 99 norm inference instances. When dynamic HL was used by the agent, it inferred the norm in 28 inference instances. For larger values of HL there isn't a significant difference between static and adaptive HL. This is because for large HL values the agent would have collected enough evidence from observing the other agents regarding the norm (i.e. the evidence of a sanction). For smaller HL values, the agent does not have enough evidence regarding the sanction. Hence, dynamically adapting the history length produces better results for an agent.



	HL=5	HL=10	HL=15	HL=20	HL=25	HL=50
■ Static HL	99	36	4	3	3	2
▨ Dynamic HL	28	23	4	3	3	2

Figure 18. Static vs. adaptive history

5.24 Similar to varying the history length dynamically, an adaptive agent can also vary its visibility threshold, the number of referrals, norm inference thresholds and the window size for identifying norms. The effects of changing these parameters have been reported in experiments 2 to 4.

Experiment 6 - Identifying dynamic norm change

5.25 An agent should have the ability to dynamically add newly identified norms and remove norms that do not hold. This experiment demonstrates that norms can be added, removed and modified by an agent dynamically depending upon the environmental conditions. The ability to change norms is important for an adaptive agent so that it can flexibly adopt norms. An agent, on identifying a norm, evaluates whether the norm holds at regular intervals of time. If the norm does not hold, it removes the norm from its norm base. Figure 19 shows the results of dynamic norm change in a society. There are two lines in the graph. The top line shows the proportion of agents in the society with a norm at any given iteration. The line that appears in the bottom shows whether punishers are present in the society. A (dummy) value of 5 means that there are punishers in the society and a value of 0 means that the punishers are not present in the society. In this experiment, the punishers do not punish[17] from iterations 300 to 600. In this experiment, having found a norm, an agent checks for the validity of the norm once again after 50 iterations. If the norm is found again, then the agent does not delete the norm. If the norm is not found, it removes the norm from its norm base. When the punishers do not punish, the norm is not inferred. As the norm is not inferred, the agent removes the norm. It can be observed that the agents start losing the norm from iteration 400 and all the agents in the society have successfully removed the norm by iteration 500. In iteration 700 some of the agents have identified the norm again and all the agents have identified the norm in iteration 850[18].

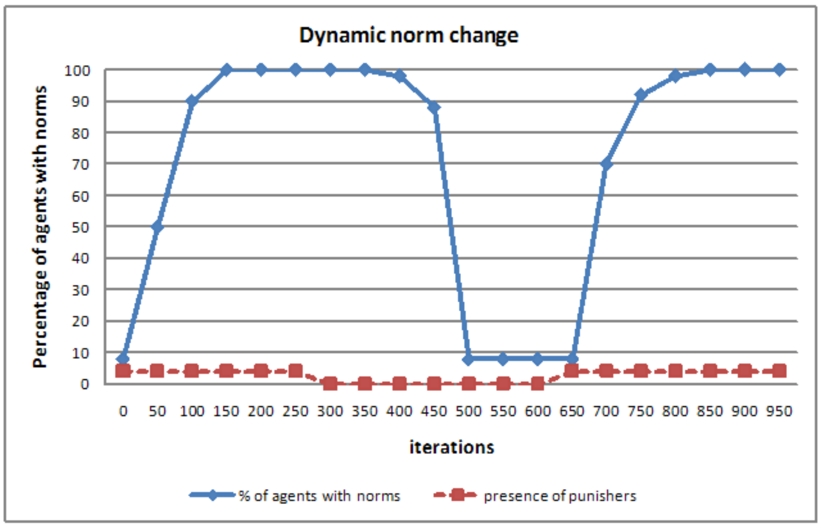


Figure 19. Dynamic norm change in an agent society

Discussion

6.1 The main contributions of the paper are the following.

1. The issue of norm identification has not been dealt with by many researchers in the field of normative multi-agent systems. To this end, in this paper we have demonstrated how one type of norm - obligation norms can be identified by an agent using the Obligation Norm Identification (ONI) algorithm. For this purpose we have made use of the norm identification architecture (Savarimuthu et al. 2010). When compared to identifying prohibition norms, identifying obligation norms is difficult because with a prohibition norm it is usually the occurrence of a particular event or a sequence of events that is the reason for a sanction to occur. In obligation norms, it is the absence of an event that is the cause of a sanction. Association rule mining algorithms only cater for the extraction of interesting sequences that are present in an event sequence. They cannot identify sequences when data items are missing. The Obligation Norm Inference (ONI) algorithm presented here can be used to generate candidate obligation norms.
2. Second, using a simple example (tipping norm identification in a restaurant), we have demonstrated how the norm inference mechanism works. In particular we have demonstrated the following:

- An agent can modify the parameters of the system based on whether it is successful in identifying a norm (e.g. the history length).
- An agent can add, remove and modify norms in a dynamically changing environment.
- An agent using this mechanism can increase the accuracy of norm identification and reduce the number of false positives generated.
- An agent is able to identify the normative protocols.

6.2 We believe the mechanism proposed in this paper can be used to identify obligation norms in several settings. For example, the norm identification architecture can be used to infer norms in Massively Multi-player Online Games (MMOGs) such as World of Warcraft (WoW). 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 proscriptions of actions (prohibition norms) and obligations that need to be satisfied (obligation norms). The mechanism proposed in this paper can be used to identify norms that are being formed. For example a norm could be that a player who has helped another player twice to escape from a dragon expects the other player to help him escape from the dragon if the need arises. This norm may not be part of the protocol defined for playing the game but may evolve during the game. Such a norm can be identified by this mechanism. Secondly, the same principle can be used in virtual environments such as Second Life to infer norms. The mechanism reported in this work can be used to identify co-existing norms (e.g. $O_{X,Y}$ (t|(e — p)) and $O_{X,Y}$ (p|(o — e))[19] can be identified). In the future we intend to extend our work to identify conflicting norms and how these conflicting norms can be handled by an agent[20].

6.3 A possible extension to this work is to allow the tipping customers to impose sanctions on non-tipping customers. We believe the impact of this addition will be the faster convergence of norms in the society. We note that the emphasis of the current work has been on norm identification.

6.4 Another potential addition to this work is on identifying conditional norms. For example, in one society, an agent may tip 10% of the bill while in another society an agent might be obliged to tip 20% of the bill. Depending upon what an agent has observed, agents may have subtly different norms. Both these agents could still infer the obligation norm but the conditions they had noticed can be different.

Conclusion

7.1 This paper addresses the question of how obligation norms can be identified in an agent society. To this end, this paper proposes the Obligation Norm Inference (ONI) algorithm. The paper uses the norm inference architecture for identifying obligation norms. An agent that employs the ONI algorithm makes use of a data mining approach to infer obligation norms. An agent can dynamically add, remove and modify norms and also can adaptively vary parameters of the system in order to identify norms. Experimental results in the context of a virtual restaurant scenario have been discussed.

Appendix

Acronyms and expansions

Table A: Acronyms used and the corresponding expansions

Acronym	Expansion
HL	History Length
ES	Event Sequences
SES	Special Event Set
UES	Unique Event Set
WS	Window Size
NIT	Norm Identification Threshold
NIF	Norm Inference Frequency
ONI	Obligation Norm Inference
EEL	Event Episode List
NEEL	Normal Event Episode List
SEEL	Special Event Episode List
NREEL	Norm Related Event Episode List
tempEEList	temporary Event Episode List
CONL	Candidate Obligation Norm List

Notes

[1] We assume that sanctioning events such as an agent yelling at another agent for violating a norm or an agent publicly shaming another agent are observable. We note that recognizing and categorizing a sanctioning event is a difficult problem. In this architecture it is assumed that such a mechanism exists (e.g. based on an agent's past experience).

[2] For example, in Massively Multi-Player Online Role Playing Games (MMORPGs), the logs of user interactions may be available for the observer through chat channels (c.f. Boella et al. 2008).

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

[4] A subsequence is a sequence that can be generated from a sequence by removing certain elements from the sequence without altering the order of the elements in the sequence.

For example, "anna" is a subsequence of "banana". Conversely, one of the supersequences of "anna" is "banana".

[5] A sub-episode is a subsequence of an event episode.

[6] A previous work (Savarimuthu et al. 2010) has shown how the former can be identified.

[7] We note that the cost of punishment is not modelled in this work because our main focus is to model and experiment with how an agent is able to recognize a norm in the first place. The cost of punishment has been experimented with in other works (Savarimuthu et al. 2008; Savarimuthu et al. 2010).

[8] We note this decision is domain specific. In some other domains such as an auction, it could be that an agent is prohibited from buying three consecutive items of the same type. In those cases each action of the same type should be recorded. We note that the mechanism proposed in this paper can handle this scenario.

[9] http://www.youtube.com/watch?v=lgBZBUbu-qg

[10] http://www.openabm.org/model-archive/norm_identification

[11] An agent may initially set the visibility threshold to a lower value so that it does not have to process a large amount of information. If it does not find a norm, it can then choose to observe interactions that happen in a larger area by increasing its visibility.

[12] When the number of referees increases, the rate of norm establishment increases. This has also been reported in many other works in multi-agent systems (Yu and Singh 2002;Yolum and Singh 2003;Candale and Sen 2005).

[13] A norm inference instance is related to NIF. An agent infers a norm once every x iterations as governed by NIF. When an agent invokes its norm inference component this is known as a norm inference instance.

[14] For example, when WS=1, if p precedes a sanction, then p-t may be identified as the protocol. In the ONI algorithm, the size of the super-sequence of a subsequence of size n is n+1.

[15] When WS is set to 5, the length of the sub-sequences can vary from one to five.

[16] PT is p-t.

[17] There can be several reasons why punishers may stop punishing. The punishers can move from one society to another or can just stop punishing because their utility has gone below a certain threshold.

[18] The simulation video can be found at http://www.youtube.com/watch?v=sZhsfliW83g.

[19] In fact, paying after eating is related to a protocol than a norm. For arguments sake, let us consider this to be a norm.

[20] We note that norm conflict resolution is being studied by some researchers (e.g. Kollingbaum et al. 2007,Vasconcelos et al. 2009).

References

AGRAWAL, R. & Srikant, R. (1994) Fast algorithms for mining association rules in large databases. In Proceedings of 20th International Conference on Very Large Data Bases (VLDB'94), Santiago de Chile, Chile (Bocca, J. B., Jarke, M. & Zaniolo, C., eds.) Morgan Kaufmann, pp. 487-499.

AKERLOF, G. A. (1976) The economics of caste and of the rat race and other woeful tales. *The Quarterly Journal of Economics* 90(4), 599-617. [doi:10.2307/1885324]

ALDEWERELD, H., Dignum, F., García-Camino, A., Noriega, P., Rodríguez-Aguilar, J. A. & Sierra, C. (2006) Operationalisation of norms for usage in electronic institutions. In: Proceedings of the fifth international joint conference on Autonomous Agents and MultiAgent Systems (AAMAS'06) New York, NY, USA: ACM Press, pp. 223-225. [doi:10.1145/1160633.1160669]

ANDRIGHETTO, G., Conte, R., Turrini, P. & Paolucci, M. (2007) Emergence in the loop: Simulating the two way dynamics of norm innovation. In: *Normative Multi-agent Systems* (Boella, G., van der Torre, L. & Verhagen, H., eds.), no. 07122 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, <http://drops.dagstuhl.de/opus/volltexte/2007/907/>

ARCOS, J. L., Esteva, M., Noriega, P., Rodríguez-Aguilar, J. A. & Sierra, C. (2005) Environment engineering for multiagent systems. *Engineering Applications of Artificial Intelligence* 18(2), pp. 191-204. [doi:10.1016/j.engappai.2004.11.019]

AXELROD, R. (1986) An evolutionary approach to norms. *American Political Science Review* 80(4), pp. 1095-1111. [doi:10.2307/1960858]

BANDURA, A. (1977) *Social Learning Theory*. General Learning Press.

BECKER, G. S. (1978) *The Economic Approach to Human Behavior*. University of Chicago Press.

BOELLA, G., Torre, L. & Verhagen, H. (2008) Introduction to the special issue on normative multiagent systems. *Autonomous Agents and Multi-Agent Systems* 17(1), pp. 1-10. [doi:10.1007/s10458-008-9047-8]

BOELLA, G. & van der Torre, L. (2006) 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) New York, NY, USA: ACM Press. [doi:10.1145/1160633.1160671]

BOMAN, M. (1999) Norms in artificial decision making. *Artificial Intelligence and Law* 7(1), pp. 17-35. [doi:10.1023/A:1008311429414]

CANDALE, T. & Sen, S. (2005) Effect of referrals on convergence to satisficing distributions. In: Proceedings of the fourth international joint conference on Autonomous Agents and Multi-Agent Systems (AAMAS'05), New York, USA, ACM Press. pp. 347-354. [doi:10.1145/1082473.1082526]

CASTELFRANCHI, C., Conte, R. & Paolucci, M. (1998) Normative reputation and the costs of compliance. Journal of Artificial Societies and Social Simulation, 1(3)3 <http://jasss.soc.surrey.ac.uk/1/3/3.html>

CEGLAR, A. & Roddick, J. F. (2006) Association mining. *ACM Computing Surveys* 38 (2), 1-42. [doi:10.1145/1132956.1132958]

CONTE, R. & Castelfranchi, C. (1999) From conventions to prescriptions - towards an integrated view of norms. *Artificial Intelligence and Law* 7(4), pp. 323-340. [doi:10.1023/A:1008310107755]

CONTE, R. & Dignum, F. (2001) From social monitoring to normative influence. *Journal of Artificial Societies and Social Simulation*, 4(2)7, <http://jasss.soc.surrey.ac.uk/4/2/7.html>

CONTE, R. & Paolucci, M. (2001) Intelligent social learning. *Journal of Artificial Societies and Social Simulation*, 4(1)3, <http://jasss.soc.surrey.ac.uk/4/1/3.html>

ELSTER, J. (1989) Social norms and economic theory. *Journal of Economic Perspectives* 3(4), pp. 99-117. [doi:10.1257/jep.3.4.99]

EPSTEIN, J. M. (2001) Learning to be thoughtless: Social norms and individual computation. *Computational Economics* 18(1), 9-24. [doi:10.1023/A:1013810410243]

GARCÍA-CAMINO, A., Rodríguez-Aguilar, J. A., Sierra, C. & Vasconcelos, W. (2006) Norm-oriented programming of electronic institutions. In: Proceedings of the fifth international joint conference on Autonomous Agents and MultiAgent Systems, AAMAS. New York, NY, USA: ACM Press. [doi:10.1145/1160633.1160750]

JONES, A. J. I. & Sergot, M. J. (1996) A formal characterisation of institutionalised power. *Logic Journal of the IGPL* 4(3), pp. 427-443. [doi:10.1093/jigpal/4.3.427]

KOLLINGBAUM, M. J., Vasconcelos, W. W., García-Camino, A. & Norman, T. J. (2007) Managing conflict resolution in norm-regulated environments. In: Proceedings of Engineering

Societies in the Agents World (ESAW'07), pp. 55-71

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

LÓPEZ y López, F., Luck, M. & d'Inverno, M. (2002) Constraining autonomy through norms. In Proceedings of The First International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS'02), pp. 674 - 681. [doi:10.1145/544862.544905]

LÓPEZ y López, F. & Marquez, A. A. (2004) An architecture for autonomous normative agents. In Proceedings of the Fifth Mexican International Conference in Computer Science (ENC'04) Los Alamitos, CA, USA: IEEE Computer Society, pp. 96-103. [doi:10.1109/enc.2004.1342594]

MANNILA, H., Toivonen, H. & Inkeri Verkamo, A. (1997) Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery* 1(3), 259-289. [doi:10.1023/A:1009748302351]

NEUMANN, M. (2010) A Classification of Normative Architectures. In Shu-Heng Chen, Claudio Cioffi-Revilla, Nigel Gilbert, Hajime Kita, Takao Terano, Keiki Takadama, Claudio Cioffi-Revilla, and Guillaume Deffuant (Eds.), *Simulating Interacting Agents and Social Phenomena, volume 7 of Agent-Based Social Systems* , Springer, pp. 3-18. [doi:10.1007/978-4-431-99781-8_1]

RYMASZEWSKI, M., Au, W. J., Wallace, M., Winters, C., Ondrejka, C., Batstone-Cunningham, B. & Rosedale, P. (2006) *Second Life: The Official Guide* . Alameda, CA, USA: SYBEX Inc.

SAVARIMUTHU, B. T. R., Purvis, M. A. & Purvis, M. K. (2008) Social norm emergence in virtual agent societies. In: Proceedings of the 7th international joint conference on Autonomous Agents and MultiAgent Systems (AAMAS '08) Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, pp. 1521-1524.

SAVARIMUTHU, B. T. R. & Cranefield, S. (2009) A categorization of simulation works on norms. In: Normative Multi-Agent Systems (Boella, G., Noriega, P., Pigozzi, G. & Verhagen, H., eds.), no. 09121 in Dagstuhl Seminar Proceedings. Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, <http://drops.dagstuhl.de/opus/volltexte/2009/1905/>

SAVARIMUTHU , B. T. R., Cranefield, S., Purvis, M. & Purvis, M. (2009) Internal agent architecture for norm identification. In: Proceeding of the international workshop on Coordination, Organization, Institutions and Norms in agent systems (COIN@AAMAS 2009), pp. 156-172.

SAVARIMUTHU, B. T. R., Cranefield, S., Purvis, M. A. & Purvis, M. K. (2010) Norm identification in multi-agent societies. Discussion Paper 2010/03, Department of Information Science, University of Otago, <http://eprints.otago.ac.nz/873/>

SCHEVE, C., Moldt, D., Fix, J. & Luede, R. (2006) My agents love to conform: Norms and emotion in the micro-macro link. *Computational and Mathematical Organization Theory* 12(2-3), pp. 81-100. [doi:10.1007/s10588-006-9538-6]

SEN, S. & Airiau, S. (2007) Emergence of norms through social learning. In: Proceedings of Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07), AAAI Press, pp. 1507-1512.

SHOHAM, Y. & Tennenholtz, M. (1992) Emergent conventions in multiagent systems: Initial experimental results and observations. In: Proceedings of third International Conference on Principles of Knowledge Representation and Reasoning. San Mateo, CA: Morgan Kaufmann, pp. 225-231.

STALLER, A. & Petta, P. (2001) Introducing emotions into the computational study of social norms: A first evaluation. *Journal of Artificial Societies and Social Simulation* , 4(1)2 <http://jasss.soc.surrey.ac.uk/4/1/2.html>

STOUP, P. (2008) The development and failure of social norms in second life. *Duke Law Journal* 58(2), pp. 311-344.

ULLMANN-MARGALIT, E. (1977) *The Emergence of Norms*. Clarendon Press.

VASCONCELOS, W. W., Kollingbaum, M. J. & Norman, T. J. (2009) Normative conflict resolution in multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 19(2), pp. 124-152. [doi:10.1007/s10458-008-9070-9]

VÁZQUEZ-SALCEDA, J. (2003) Thesis: The role of norms and electronic institutions in multi-agent systems applied to complex domains, the harmonia framework, *AI Communications* 16(3), pp. 209-212.

VERHAGEN, H. (2001) Simulation of the Learning of Norms. *Social Science Computer Review* 19(3), pp. 296-306. [doi:10.1177/089443930101900305]

WALKER, A. & Wooldridge, M. (1995) Understanding the emergence of conventions in multi-agent systems. In: Proceedings of the First International Conference on Multi-Agent Systems (Lesser, V., ed.) San Francisco, CA: MIT Press, pp. 225-231.

Weisstein, E. W. (2010) Moore neighbourhood, < http://mathworld.wolfram.com/MooreNeighborhood.html>

YOLUM, P. & Singh, M. P. (2003) Emergent properties of referral systems. In Proceedings of the Autonomous Agents and Multi-agent Systems (AAMAS 03), pp. 592-599. [doi:10.1145/860575.860670]

YU, B. & Singh, M. P. (2002) Emergence of agent-based referral networks. In Proceedings of the Autonomous Agents and Multi-agent Systems (AAMAS 02), pp. 1268-1269. [doi:10.1145/545056.545113]