A data mining approach to identify obligation norms in agent societies

Bastin Tony Roy Savarimuthu, Stephen Cranefield, Maryam Purvis and Martin Purvis

University of Otago, Dunedin, P O Box 56, Dunedin, New Zealand
(tonyr,scranefield,tehrany,mpurvis)@infoscience.otago.ac.nz

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 how an agent can make use of the proposed norm identification architecture to identify norms. This paper explains how an agent using this architecture identifies one type of norm, an obligation norm. To this end, the paper proposes an Obligation Norm Inference (ONI) algorithm which makes use of association rule mining approach to identify obligation norms.

1 Introduction

Most works on norms in normative multi-agent systems have concentrated on how norms regulate behaviour (e.g. [10]). 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 [7] or through an institution that prescribes what the norms of the society should be [19].

Only a few researchers have dealt with how an agent may infer what the norms of a newly joined society are [2]. 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 [14] 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 [18]. When a community of agents determines what the norm should be, the norm can evolve over time. So, a new agent joining the society should have the ability to recognize the changes to the norms.

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 obligation norms in a society using the Obligation Norm Inference (ONI) algorithm presented here. Using an auction example, we demonstrate how an agent makes use of the norm identification framework.

The paper is organized as follows. Section 2 provides a background on normative multi-agent systems (NorMAS). Section 3 provides an overview of the norm identification framework. Section 4 describes the components of the framework in the context of an e-market scenario (buying and selling goods). Section 5 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 6.

2 Background

Due to multi-disciplinary interest in norms, several definitions for norms exist [2]. The definition of normative multi-agent systems as described by the researchers involved in the NorMAS 2007 workshop is as follows [6]. *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 [17, 20].

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. Several architectures have been proposed for normative agents (refer to [12] for an overview). 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 (refer to [15].

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 [4, 17]. 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 [3]. Very few have investigated how an agent comes to know the norms of the society.

Our objective in this work are two-fold. First, we propose an architecture which can be used by individual agents to identify what the norms of the society are. The architecture is based on observation of agent interactions and the inference mechanism considers "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 may not only imply a monetary punishment, it could also be an action that could invoke emotions (such as an agent yelling at another potentially invoking shame or embarrassment on another agent), which can help in norm spreading. Agents can recognize

such actions based on their previous experience. Second, based on association rule mining [9], 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. The ONI algorithm identifies potential obligation norms. An example obligation for an agent participating in an online auction is to pay for the item it has won ($O_{A,B}(p|w)$)[1]. When obligations are violated, sanctions can be imposed on the violating agent by other agents. In this architecture we assume that agents have the ability to recognise sanctions. The rest of the paper describes how an observer agent will be able to infer a norm of the society.

3 Overview of the norm identification architecture

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. The norm identification framework takes into account the social learning theory [5] that suggests that new behavior can be learnt through the observation of punishment and rewards. 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 four-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 an auction, a regular event is an event, such as an agent bidding for an item and winning the item. Special events are signalling events that agents understand to be either encouraging or discouraging certain behaviour. For example when an agent wins a particular item but does not pay in accordance to the norm of the society, the agent can be sanctioned by the auction authorities[2]. Let us assume that the signal in this context is the occurrence of the shaming event which is a form of a sanction. In this work 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

[1] This is to be read as A is obliged to B to bring about *p* given *w* has occurred.

[2] For example, when the winner who is obliged to submit a cheque immediately after winning does not submit the auctioneer may denounce the winner, black list the winner or even report it to authorities such as the Police.

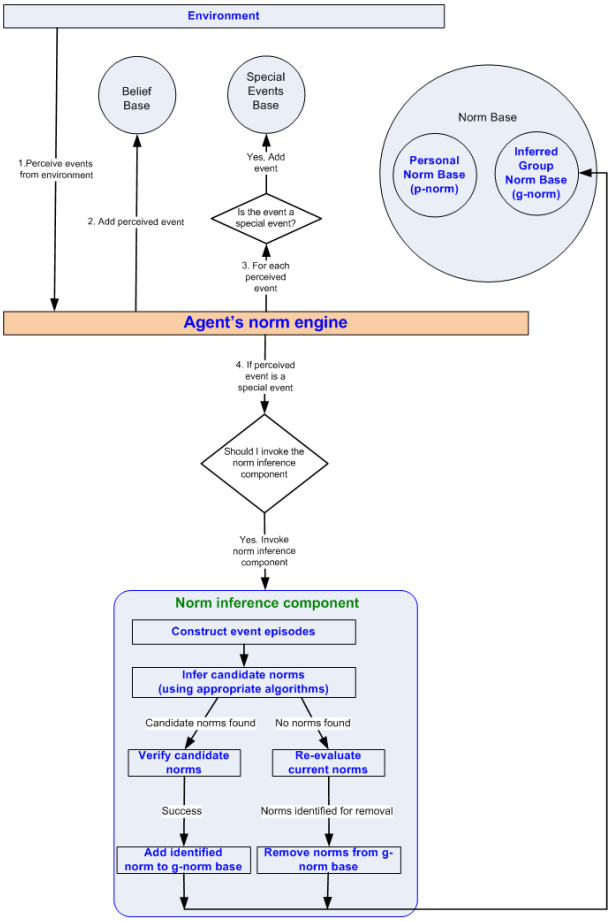

Fig. 1: Norm identification architecture of an agent

may possess some p-norms[3] 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 personal 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 the auction scenario, an agent observing a sanctioning event may invoke its norm inference component to find out what events that

[3] A *p-norm* is the personal value of an agent. For example an agent may consider that bidding for an item of the same type that the bidder has won previously is an action that should be prohibited in a society. This personal value may not be shared by the agents in a society.

had happened in the past (or that had not happened in the past) may have triggered the occurrence of the special event[4]. 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.

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 4.3. What an agent does with the norms once it has inferred the norms is out of the scope of this paper.

4 Obligation norm identification

In this section we explain how obligation norms can be identified using the framework described in Section 3. Obligation norms can be identified by an agent in our architecture using obligation norm inference (ONI) algorithm proposed here. First, we describe the domain in which an obligation norm is identified. Second, we describe the event storage components of the architecture (steps 2 and 3 of Figure 1). Third, we describe the ONI algorithm.

4.1 e-market scenario

Let us assume that agents participate in an electronic market such as an auction in a virtual environment (e.g. Second Life). A new agent joining a society may not be aware of the norms associated with buying and selling goods in an electronic market. For example, in one society the winning bidder may be obliged to deposit the money within a day while the norm in another society could be that the winner may be obliged to deposit the money within an hour. Failure to fulfill the obligation may result in a sanction.

4.2 Event storage components

Let us assume that an agent is situated in an auction house where multiple auctions take place at the same time (the electronic market scenario [13]). Let us also assume that a new agent is not aware of the norms of the auction house. In this architecture an agent would first observe the interactions that occur between the agents in the society. The

[4] Prohibition norms may be identified by inferring the relevant events that happened in the past. For identifying obligation norms the agent may have to reason about what events that did not happen in the past are the likely reason for a sanction (i.e. not fulfilling an obligation)

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. bidding). 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).

In the auction scenario, the agent is aware of the actions performed by an agent, which are bidding for an item (*bid*), winning an item (*win*), paying for a particular item (*pay*) and receiving the item (*receive*). The agent also has the ability to recognize a signalling action such as *yell* or *shame*[5]. Signalling events can either be positive (e.g. rewards) or negative (sanctions)[6].

Let us assume that a new agent (an observer) is situated in the auction. The observer records interactions that occur in the context of the auction. 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.

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 bidding will represent this as ***happens(1,bid(X,item1,Y)***. This implies the observer believes that the first event was generated by agent X which bids for item1 to agent Y. A sample representation of events observed by an agent is given in list (1). An agent situated in an environment can sense these actions through observation or through action logs that may be available[7].

$$\begin{pmatrix} happens(1, bid(C, item1, B) \\ happens(2, bid(A, item1, B) \\ happens(3, win(A, item1, B) \\ happens(4, bid(A, item2, C) \\ happens(5, win(A, item2, C) \\ happens(6, sanction(B, A)) \\ happens(7, pay(A, item2, C) \\ happens(8, receive(A, item2, C) \end{pmatrix} \tag{1}$$

An agent records these events in its belief base. Event 6 is a sanctioning event, where agent B sanctions agent A. The reason for the sanction is that agent A failed to pay for

[5] We assume that sanctioning events such as an agent yelling at another agent for violating a norm or an agent publicly shaming another agent by announcing that the agent is blacklisted are observable. We note that 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)

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

[7] 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 [6]

the item it has bought. For an observer (the agent) 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 such as open e-markets, 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.

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 under the *sanction* event.

4.3 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 Obligation Norm Inference (ONI) algorithm to generate candidate obligation norms. Candidate obligation 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 ONI algorithm. Secondly we describe the internal details of ONI algorithm using the auction example.

4.3.1 Definitions of parameters used in the algorithm The parameters that are used in the Obligation Norm Inference algorithm are explained below.

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.

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 list (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 event set can contain events such as yell (SES = $\{yell\}$). An agent also has the capability to categorize events into two types, sanctions and rewards. For example the action mentioned above can be identified as a sanctioning action.

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 list (1) contains the following events[8], UES = $\{bid,win,pay,receive,sanction\}$.

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

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

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

Window size (WS): When an agent wants to infer norms, it looks into its history for a certain number of recent events. For example, if the WS is set to 3, an agent constructs 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 buyer and the seller). Construction of event episodes is described in the next sub-section. It should be noted that an EE is a subsequence[9] 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[10] of an event episode that have a 50% chance of being a candidate norm. The algorithm uses the NIT on three occasions. As the values of NIT for each of the occasions can be varied by an agent, there are three variables which are NIT_a, NIT_b and NIT_c.

Norm Inference Frequency (NIF): An agent may choose to invoke a norm inference component every time it observes a special event, or it may invoke the component periodically. An agent has a parameter called the 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.

4.3.2 Obligation Norm Inference (ONI) algorithm

4.3.2.1 Overview of the algorithm

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 that are exchanged between agents. 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 contains all event episodes that contain at least one sanctioning event called the Special Event Episode List (SEEL). The second list contains all event episodes that do not contain sanctioning events called the Normal Event Episode List (NEEL). Third, using SEEL, all sub-episodes which have occurrence probabilities greater than or equal to NIT_a are extracted and stored in Norm-Related Event Episode List (NREEL) based on a modified version of the WINEPI algorithm [11]. Fourth, for each event episode in NREEL, all supersequences whose occurrence probabilities are greater than or equal to

[9] 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”.

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

NIT_b are extracted 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 whose occurrence probabilities are greater than or equal to NIT_c which are stored in Candidate Obligation Norm List (CONL). These four steps are explained in detail in the following sub-sections.

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) using NEEL and NREEL ;
      /* Algorithm 2 */
6 end
```

4.3.2.2 Creating event episodes

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 three agents A, B and C, participating in an auction as given in list (1) and an observer D. Agent C bids for item1 from agent B. Agent A bids for item1 from agent B. Agent A wins item1. Agent A then bids for item2 from agent C. Agent A wins item2. Agent A is sanctioned by agent B. Agent A pays agent C for item2. Agent A receives item2 sent by agent C[11]. 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 *n* events that happened between a pair of agents from the recorded history (event sequences (ES)) where *n=WS*. We call the retrieved event sequence the *event episode* (EE). A sample event episode from an observer's view-point (agent D) is given below. The left hand side of the arrow indicates that the agents involved in the interaction are A and B. The right hand side of the arrow contains the event episode. A hyphen separates one event from the next.

$$\{A, B\} \rightarrow (happens(2, bid(A, item1, B) - happens(3, win(A, item1, B) - happens(6, sanction(B, A)))$$

Based on the what an agent observes (e.g. the event sequence given in list (1)), 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 certain action(s) might have caused a sanction. In this work we concentrate on the latter[12]. Agent D then extracts the sequence of events (the *event episode*) that took place between A and B based on the event sequence stored in its history. To simplify the notation here, only the first letter of each event will be mentioned from here on (e.g. *b* for *bid*) and also the agent names

[11] Note that the representation of these actions are from one agent's point of view (e.g. agent A's point of view). Therefore actions such as *send* which is from the viewpoint of agent C are not modelled, but a related action i.e. *receive* is modelled from A's view-point.

[12] In previous work we have demonstrated how the former can be identified [16].

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 B shown above will be represented as

$$\left(\{A, B\} \rightarrow b - w - s \right)$$

The following list (2) shows a sample event episode list (EEL) that contains ten events occurring between a pair of agents that are observed by an agent where WS=5. Note that the Unique Event Set (UES) in this case include events $\{b,w,p,r,s\}$ which stand for $\{$*bid,win,pay,receive,sanction*$\}$ respectively.

$$\begin{pmatrix} \{A, B\} \rightarrow (b - w - s) \\ \{C, D\} \rightarrow (w - p - r) \\ \{E, F\} \rightarrow (b - w - p - r - b) \\ \{G, H\} \rightarrow (p - r - b - w) \\ \{I, J\} \rightarrow (r - b - b - w - s) \\ \{K, L\} \rightarrow (b - w - p) \\ \{M, N\} \rightarrow (r - b - w - p - r) \\ \{O, P\} \rightarrow (b - w - p - r) \\ \{R, S\} \rightarrow (r - b - w - s) \\ \{T, U\} \rightarrow (r - b - w - p) \end{pmatrix} \tag{2}$$

4.3.2.3 Creating special and normal event episode lists

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).

The SEEL obtained from EEL is given in the left in (3). NEEL has the remaining episodes that do not contain a sanctioning action (shown in the right of (3)).

$$\begin{pmatrix} (b - w - s) \\ (r - b - b - w - s) \\ (r - b - w - s) \end{pmatrix}, \begin{pmatrix} (w - p - r) \\ (b - w - p - r - b) \\ (p - r - b - w) \\ (b - w - p) \\ (r - b - w - p - r) \\ (b - w - p - r) \\ (r - b - w - p) \end{pmatrix} \tag{3}$$

4.3.2.4 Generating the norm related event list (NREEL)

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 (3), the agent may infer that the sub-episodes *b-w*, *b*, or *w* 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 (reported in previous work [16]). In the case of an obligation norm, the agent first has to find the frequently occurrence 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 whose occurrence probability is greater than or equal to NIT_a, which is 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.

In order to identify these norm-related events the agent uses a modified version of the WINEPI algorithm [11], an association rule mining algorithm[13]. The modification, reported in previous work [16], can identify candidate norms that are obtained by considering "permutations with repetition" when constructing sub-episodes. Based on the SEEL, an agent can generate the NREEL. Expression (4) shows the SEEL on the left of the arrow and the NREEL generated from the SEEL on the right of the arrow when NIT_a is set to 0. The occurrence probability of an event episode in NREEL is given in square brackets. When NIT_a is set to 0, all possible subsequences of event episodes in SEEL are generated. When NIT_a is set to 100% the algorithm identifies the following norm-related event episode list $\{$*b*-*w*,*b*,*w*$\}$. An agent, being an autonomous entity, can vary the NIT_a parameter to identify the norm-related events. Note that if an event episode is frequent, all its subsequences are also frequent. For example if *b*-*w* appears 100% of the time (i.e. the occurrence probability is 1), all its subsequences also appear 100% of the time.

$$\begin{pmatrix} (b-w-s) \\ (r-b-b-w-s) \\ (r-b-w-s) \end{pmatrix} \rightarrow \begin{pmatrix} (b-w)[1] \\ (b)[1] \\ (w)[1] \\ (r-b-w)[.66] \\ (r-b)[.66] \\ (r-w)[.66] \\ (r)[.66] \\ (r-b-b-w)[.33] \\ (r-b-b)[.33] \\ (b-b-w)[.33] \\ (b-b)[.33] \end{pmatrix} \tag{4}$$

4.3.2.5 Identifying candidate obligation norm list (CONL)

[13] Association rule mining [9] 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) [1, 11]. A limitation of the well-known Apriori [1] algorithm is that it considers combinations of events but not permutations (e.g. it does not distinguish between event sequences *b*-*w* and *w*-*b*). WINEPI [11] 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. *b* and *w*, the algorithm can generate sub-episodes of length two which are *bw* and *wb*, but not *bb* and *ww*). Permutations with repetition are important because there could be a norm which sanctions an agent from performing the same action twice.

The pseudo code 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 whose occurrence probabilities are greater than or equal to NIT_b. These supersequences are stored in a list (*tempEEList* in this case).

Based on the supersequences stored in *tempEEList*, the algorithm (previous work [16]) can identify all permutations of supersequences whose occurrence probabilities are greater than or equal to NIT_c. Such supersequences are stored in the candidate obligation norm list (CONL).

For example, let us consider an event epsiode *b*-*w* is the only event episode stored in the NREEL list. Assume this NREEL is the input to Algorithm 2 and the NIT_b is set to 50%. Expression (5) 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 seven event episodes but *tempEEList* contains six out of seven event episodes that contain *b*-*w*. These six event episodes are supersequences of *b*-*w*.

Algorithm 2: Pseudocode to create 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 = ∅;
2  for an event episode NREE ∈ NREEL do
3      tempEEList= ∅;
4      tempCounter = 0;
5      occurrenceCounter = 0;
6      foreach event episode EE ∈ NEEL do
7          occurrenceCounter++;
8          if EE is a supersequence of NREE then
9              Add EE to tempEEList;
10             tempCounter++;
11         end
12     end
13     OP(tempEEList) = tempCounter/occurrenceCounter;
14     if OP(tempEEList) ≥ NIT_b then
15         Use modified WINEPI algorithm to extract all candidate obligation
           norms (Input: tempEEList, Unique Event Set (UES), Window Size(WS),
           Norm Inference Threshold (NIT_c), Output: Candidate norms);
16         Add candidate obligation norms to CONL;
17     end
18     return CONL;
19 end
```

$$\begin{pmatrix} (w-p-r) \\ (b-w-p-r-b) \\ (p-r-b-w) \\ (b-w-p) \\ (r-b-w-p-r) \\ (b-w-p-r) \\ (r-b-w-p) \end{pmatrix} \rightarrow \begin{pmatrix} (b-w-p-r-b) \\ (p-r-b-w) \\ (b-w-p) \\ (r-b-w-p-r) \\ (b-w-p-r) \\ (r-b-w-p) \end{pmatrix} \quad (5)$$

From *tempEEList* the CONL can be generated. The left hand side of expression (6) shows the *tempEEList*. The right hand side of expression (6) contains all permutations of supersequences of *b-w* that can be obtained from *tempEEList* and their occurrence probabilities in *tempEEList* (in square brackets).

$$\begin{pmatrix} (b-w-p-r-b) \\ (p-r-b-w) \\ (b-w-p) \\ (r-b-w-p-r) \\ (b-w-p-r) \\ (r-b-w-p) \end{pmatrix} \rightarrow \begin{pmatrix} (b-w-p)[0.83] \\ (b-w-r)[0.5] \\ (b-w-p-r)[0.5] \\ (r-b-w)[0.5] \\ (b-w-b)[0.16] \\ (b-w-p-b)[0.16] \\ (b-w-p-r-b)[0.16] \\ (p-r-b-w)[0.16] \\ (p-b-w)[0.16] \end{pmatrix} \quad (6)$$

Assuming that NIT_c is set to 50%, the supersequences that will be identified as CONL are *(b-w-p,b-w-r,b-w-p-r,r-b-w)* whose occurrence probabilities are *(0.83,0.5,0.5,0.5)* respectively. As the occurrence probabilities of *(b-w-b,b-w-p-b,b-w-p-r-b,p-r-b-w,p-b-w)* are less than NIT_c, these are not included in the CONL. Note that the modified WINEPI algorithm is used twice, 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 15 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 subsection.

4.4 Norm verification

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 *b-w-p* for which it has a higher occurrence probability) and asks B if it knows whether the obligation norm $O_{X,Y}(p|(b-w))$ is a norm of the society (i.e. an agent is obliged to pay after bidding and winning). 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[14].

[14] 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.

In the case of the running example, the supersequence *b-w-p* 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 $O_{X,Y}(p|(b-w))$ 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 the failure to fulfill the obligation norm $O_{X,Y}(r|(b-w))$ is the reason for the sanction. Otherwise, the next event episode in the candidate norm list is chosen for verification. This process continues until a norm is found or no norm is found from the event episodes in the candidate norm list. If no norm is found, the agent considers the next event episode in the NREEL and uses Algorithm 2 to identify candidate obligation norms. This process continues until there are no event episodes in the NREEL. Even in the case of identifying a candidate norm, the agent continues the process to identify any co-existing norms.

Note that an agent will have two sets of norms: candidate norms and identified norms. Expression (7) shows the two sets of norms, the candidate norms on the left of the arrow and the identified norms 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).

$$\begin{pmatrix} (b-w-p) \\ (b-w-r) \\ (b-w-p-r) \\ (r-b-w) \end{pmatrix} \rightarrow \big(\, b-w-p \,\big) \tag{7}$$

5 Discussion

The main contributions of the paper are two-fold. First, 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 proposed an architecture for norm identification and have demonstrated how one type of norm - the obligation norm can be identified by an agent. Secondly, we have proposed the Obligation Norm Inference (ONI) algorithm, an algorithm based on data mining, that can be used to generate candidate obligation norms. Using a simple example, we have demonstrated how the norm inference mechanism works. An adaptive agent employing the norm inference mechanism can infer obligation norms by varying different parameters of the algorithm.

We believe this architecture can be used in several settings apart from e-commerce environments. For example, the norm identification architecture can be used to infer norms in Massively Multi-player Online Role Playing Games (MMORPGs) 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 normative architecture 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.

An interesting addition to this work is on identifying conditional norms. For example, in one society, the norm associated with the deadline for the payment (i.e. obligations with deadlines as in [8]) may be set to 120 minutes after winning the item. Depending upon what an agent has observed, agents may have subtly different norms (e.g. one agent may notice that p follows w after an average of 100 minutes while another may notice this to happen after 150 minutes). Both these agents could still infer the obligation norm but the deadlines they had noticed can be different. Another interesting avenue for research is to investigate employing string based pattern matching algorithms used in the field of bio-informatics to extract interesting sequences and missing sequences.

6 Conclusion

This paper addresses the question of how obligation norms can be identified in an agent society. To this end, this paper uses the norm inference architecture for identifying obligation norms. This paper proposes Obligation Norm Inference (ONI) algorithm. An agent that employs ONI algorithm makes use of data mining approach to infer obligation norms. This has been demonstrated in the context of a simple e-market scenario.

References

1. Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proceedings of 20th International Conference on Very Large Data Bases (VLDB'94), Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.
2. Giulia Andrighetto, Rosaria Conte, Paolo Turrini, and Mario Paolucci. Emergence in the loop: Simulating the two way dynamics of norm innovation. In Guido Boella, Leon van der Torre, and Harko Verhagen, editors, *Normative Multi-agent Systems*, number 07122 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.
3. Josep Ll. Arcos, Marc Esteva, Pablo Noriega, Juan A. Rodriguez-Aguilar, and Carles Sierra. Environment engineering for multiagent systems. *Engineering Applications of Artificial Intelligence*, 18(2):191204, 2005.
4. Robert Axelrod. An evolutionary approach to norms. *The American Political Science Review*, 80(4):1095–1111, 1986.
5. Albert Bandura. *Social Learning Theory*. General Learning Press, 1977.
6. Guido Boella, Leendert Torre, and Harko Verhagen. Introduction to the special issue on normative multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 17(1):1–10, 2008.
7. Magnus Boman. Norms in artificial decision making. *Artificial Intelligence and Law*, 7(1):17–35, 1999.

8. Henrique Lopes Cardoso and Eugenio Oliveira. Directed deadline obligations in agent-based business contracts. In *Proceeding of the international workshop on Coordination, Organization, Institutions and Norms in agent systems (COIN@AAMAS 2009)*, 2009.
9. Aaron Ceglar and John F. Roddick. Association mining. *ACM Computing Surveys*, 38(2):5, 2006.
10. Fabiola López y López. *Social Powers and Norms: Impact on Agent Behaviour*. PhD thesis, Department of Electronics and Computer Science, University of Southampton, United Kingdom, 2003.
11. Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.
12. Martin Neumann. A classification of normative architectures. In *Proceedings of World Congress on Social Simulation*, 2008.
13. Juan A. Rodrguez-aguilar, Francisco J. Martn, Pablo Noriega, Pere Garcia, and Carles Sierra. Towards a test-bed for trading agents in electronic auction markets. *AI Communications*, 11:5–19, 1998.
14. Michael Rymaszewski, Wagner James Au, Mark Wallace, Catherine Winters, Cory Ondrejka, Benjamin Batstone-Cunningham, and Philip Rosedale. *Second Life: The Official Guide*. SYBEX Inc., Alameda, CA, USA, 2006.
15. Bastin Tony Roy Savarimuthu and Stephen Cranefield. A categorization of simulation works on norms. In Guido Boella, Pablo Noriega, Gabriella Pigozzi, and Harko Verhagen, editors, *Normative Multi-Agent Systems*, number 09121 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.
16. Bastin Tony Roy Savarimuthu, Stephen Cranefield, Maryam A. Purvis, and Martin K. Purvis. Norm identification in multi-agent societies. Discussion Paper 2010/03, Department of Information Science, University of Otago.
17. Yoav Shoham and Moshe 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*, pages 225–231, San Mateo, CA, 1992. Morgan Kaufmann.
18. Phillip Stoup. Athe development and failure of social norms in second life. *Duke Law Journal*, 58(2):311–344, 2008.
19. Javier 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):209–212, 2003.
20. Adam Walker and Michael Wooldridge. Understanding the emergence of conventions in multi-agent systems. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi–Agent Systems*, pages 384–389, San Francisco, CA, 1995. MIT Press.