Selective Norm Monitoring

Natalia Criado
King's College London, UK
natalia.criado@kcl.ac.uk

Jose M. Such
Lancaster University, UK
j.such@lancaster.ac.uk

Abstract

Real-world norm monitors have limited capabilities for observing agents. This paper proposes a novel mechanism to take full advantage of limited observation capabilities by selecting the agents to be monitored. Our evaluation shows this significantly increases the number of violations detected.

1 Introduction

Within the Multi-agent System (MAS) area, norms coordinate and regulate the activity of autonomous agents interacting in a given social context [López y López *et al.*, 2006]. Several authors have proposed infrastructures to monitor agent actions and detect norm violations [Gaertner *et al.*, 2007; Minsky and Ungureanu, 2000; Modgil *et al.*, 2009]. The majority of these proposals assumed that actions are always observable. However, this assumption does not always hold and, in practice, norm monitors may have limited observation capabilities. Very recent work on imperfect norm monitoring proposes solutions to ensure complete observability either by adding more monitors to observe all agents [Bulling *et al.*, 2013] or by adapting the norms to what can be monitored [Alechina *et al.*, 2014]. However, there are circumstances in which norms cannot be modified (e.g., contract/law monitoring) or adding monitors is expensive and/or not feasible. This paper goes beyond these approaches by predicting the actions that can be executed by each agent to select those agents that are worth monitoring, so norm monitors could focus their limited observation capabilities on them.

2 Preliminary Definitions

$\mathcal{L}$ is a first-order language containing a finite set of predicate and constant symbols, the logical connective $\neg$, the equality (inequality) symbol $=$ ($\neq$), the true (false) proposition $\top$ ($\bot$), and an infinite set of variables. The predicate and constant symbols are written beginning with a lower case letter. Variables are written beginning with a capital letter. We will relate our formulae via logical entailment $\vdash$ ($\nvdash$). We also assume the standard notion of substitution of variables [Fitting, 1996]; i.e., a substitution σ is a finite and possibly empty set of pairs Y/y where Y is a variable and y is a term.

The set of grounded atomic formulas of $\mathcal{L}$ is built of a finite set of predicates that characterise the properties of the

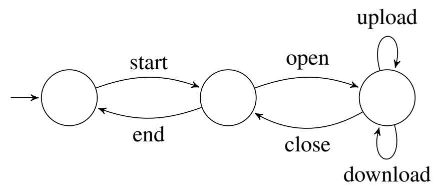

Figure 1: Network File Sharing Protocol

world relevant to norm monitoring. By a situation, we mean the properties that are true at a particular moment. Some of these properties are static and not altered by action execution, whereas other properties are dynamic and changed due to agent actions. Specifically, we represent static properties as a set[1] of atomic grounded formulas of $\mathcal{L}$, denoted by g. A state s is a set of grounded atomic formulas of $\mathcal{L}$, describing dynamic properties which hold on state s. Thus, a situation is built on a "closed world assumption" and defined by a set of static properties g and a state s.

EXAMPLE 1. *To illustrate our proposal, we shall use a simplified example of users interacting with a network file sharing system using the protocol depicted in Figure 1. In this example, $\mathcal{L}$ contains: 4 predicate symbols ($user, file, session, opened$), to represent users, files, users having initiated a session on the system, and files opened by users; and constant symbols representing users ($u1, u2, u3$) and files ($f1, f2$). Information about users and files is static and represented as:*

$$g = \{user(u1), user(u2), user(u3), file(f1), file(f2)\}$$

Information about sessions and open files is dynamic. Specifically, the initial state s_0 is defined as follows:

$$s_0 = \{session(u2), session(u3), opened(u2, f1)\}$$

Action Definitions. In line with the existing literature [Boutilier and Brafman, 2001], actions are represented using preconditions and postconditions. If a situation does not satisfy the preconditions, then the action cannot be applied. In contrast, if the preconditions are satisfied, then the action can be applied transforming the current state into a new state in which all negative (vs. positive) literals appearing in the postconditions are deleted (vs. added).

[1] In this paper sets are to be interpreted as the conjunction of their elements.

Definition 1. An *action description* is a tuple $\langle name, pre, post\rangle$ where:

- $name$ is the action name;
- pre is the precondition, i.e., a set of positive and negative literals of $\mathcal{L}$ (containing both dynamic and static properties) as well as equality and inequality constraints on the variables;
- $post$ is the postcondition; i.e., a set of positive and negative literals of $\mathcal{L}$ (containing dynamic properties only).

Given an action description d, we denote by $pre(d)$, and $post(d)$ the action precondition, and postcondition.

EXAMPLE 2. *Figure 1 shows the 6 actions*[2] *considered in our running example represented as arcs, each of which has associated the following action description:*

$$\langle start, \{user(U), \neg session(U)\}, \{session(U)\}\rangle$$
$$\langle end, \{user(U), session(U), \neg opened(U,F)\}, \{\neg session(U)\}\rangle$$
$$\langle open, \{user(U), session(U), file(F1), file(F2), \neg opened(U,F2)\}, \{opened(U,F1)\}\rangle$$
$$\langle upload, \{user(U), session(U), file(F), opened(U,F)\}, \{\}\rangle$$
$$\langle download, \{user(U), session(U), file(F), opened(U,F)\}, \{\}\rangle$$
$$\langle close, \{user(U), session(U), file(F), opened(U,F)\}, \{\neg opened(U,F)\}\rangle$$

Definition 2. Given a situation represented by a state s and a set of static properties g, and an action description $\langle name, pre, post\rangle$; an *action instance* (or action) is a tuple $\langle name, pre', post'\rangle$ such that:

- There is a substitution σ of variables in pre, such that $s, g \vdash \sigma \cdot pre$;
- pre' is a set of grounded literals in $\sigma \cdot pre$ containing dynamic properties only;
- $post' = \sigma \cdot post$.

Given an action a, we denote by $actor(a)$ the agent performing the action, and by $pre(a), post(a)$ the precondition, and postcondition.

Given a set of actions $A = \{a_1, ..., a_n\}$, we define $pre(A) = \bigcup pre(a_i)$, $post(A) = \bigcup post(a_i)$ and $actor(A) = \bigcup actor(a_i)$.

In a MAS, *concurrent actions*, which are sets of actions that occur at the same time and do not necessarily imply agent coordination, define state transitions. For the sake of simplicity, we assume that each agent performs one action at a time. We also assume that concurrent actions are mutually consistent; i.e., in a concurrent action there are not contradictions among the actions' preconditions and postconditions.

EXAMPLE 3. *Assume that at state s_0 the following concurrent action occurred — action descriptions and instances are represented by their name and parameters:*

$$A = \{start(u1), upload(u2, f1), open(u3, f1)\}$$

Norm Definitions. We consider *norms* as formal statements that define patterns of behaviour by means of *deontic modalities* (i.e., *obligations* and *prohibitions*). Specifically, we consider norms as conditional rules of behaviour that define under which circumstances a pattern of behaviour becomes relevant and must be fulfilled [López y López *et al.*, 2006; Vasconcelos *et al.*, 2007].

Definition 3. A *norm* is defined as a tuple $\langle deontic, condition, action\rangle$, where:

- $deontic \in \{\mathcal{O}, \mathcal{F}\}$ is the deontic modality, determining if the norm is an obligation ($\mathcal{O}$) or prohibition ($\mathcal{F}$);
- $condition$ is a set of literals of $\mathcal{L}$ as well as equality and inequality constraints that represents the situations in which the norm is relevant;
- $action$ is a (possibly instantiated) action description.

We consider a *closed legal system*, where everything is considered permitted by default, and obligation and prohibition norms define exceptions to this default permission rule. We define that a norm is relevant to a specific situation (instantiated) if the norm condition is satisfied in the situation; i.e., if there is a substitution of the variables in the norm condition such that the constraints in the norm condition are satisfied and the positive (vs. negative) literals in the norm condition are true (vs. false) in the situation. Finally, the semantics of instances (and norms in general) depends on their deontic modality. An obligation instance is fulfilled when the mandatory action is performed and violated otherwise, while a prohibition instance is violated when the forbidden action is performed and fulfilled otherwise.

EXAMPLE 4. *In our example, there is a norm that forbids a user to open a file when another user has already opened it to avoid concurrent access problems:*

$$\langle \mathcal{F}, \{user(U2), file(F), opened(U2,F)\}, open(U1,F)\rangle$$

which instantiated as follows in state s_0 (see Example 1):

$$\langle \mathcal{F}, open(U1, f1)\rangle \text{ where } \sigma = \{F/f1, U2/u2\}$$

That is, as $u2$ had already opened the file $f1$, then any other user $U1$ who opens the file violates the norm; e.g., $u3$ violates the norm (see Example 3).

3 NM Information Model

Definition 4. A *norm monitor* (NM) is defined as a tuple $\langle G, N, D, o\rangle$ where: G is a set of agents to be monitored; N is the set of norms that regulate agent actions; D is a set of action descriptions that represents the actions that can be performed by agents; and $o \in \mathbb{N} : o \leq |G|$ represents the observation capabilities of the monitor (i.e., the number of agents that can be monitored simultaneously). Note this o models NMs can have different observation capabilities. The evaluation section proves the significant improvements obtained by our NMs regardless of these observation capabilities.

The goal of the NM is to select the set of agents to be monitored to maximize the number of norm violations and fulfilments detected. Norm enforcement is out of the scope of this work and we assume that once the NM detects a norm violation (vs. fulfilment), it applies the corresponding sanction (vs. reward).

[2] Our proposal is agnostic wrt. the existence of a NOP action that allows agents to remain still. For the sake of clarity and simplicity, the running example does not include the NOP action. However, our extensive experiments include the NOP action (see Section 4).

3.1 State Representation

As the NM may observe a subset of the actions performed by agents, it has partial information about the state of the world. The NM represents each partial state of the world, denoted by p, using an "open world assumption" as a set of grounded literals that are known in the state. Thus, a partial state contains positive (vs. negative) grounded literals representing dynamic properties known to be true (vs. false) in the state. The rest of dynamic properties are unknown.

We assume that the NM monitor has complete knowledge of the initial state. Thus, at $t = 0$ the NM knows which grounded atomic formulas are true or false in the current state, i.e., the *partial* state p_0 is equivalent to s_0.

EXAMPLE 5. *In our example, the NM knows which grounded atomic formulas are true or false in the initial state ($p_0 \equiv s_0$):*

$$p_0 = \{\neg session(u1), session(u2), session(u3) \\ \neg opened(u1, f1), opened(u2, f1), \neg opened(u3, f1), \\ \neg opened(u1, f2), \neg opened(u2, f2), \neg opened(u3, f2)\}$$

3.2 Action Prediction

As defined above, the NM has limited capabilities for observing agent actions, so it should decide which agents will be monitored in the next step to make the most of its capabilities. In this paper, we propose that the NM makes this decision based on the actions that can be performed by agents according to the current state of the word. In particular, the NM predicts the actions agents may execute and ranks agents according to the chances of violating/fulfilling a norm in the next time step. In the following, we introduce full and approximate methods for predicting agent actions.

Full Action Prediction. Full prediction searches exhaustively the actions that can be performed by all the agents.

Definition 5. Given a partial state description p (current state); we define *search* as a function that computes sets of solutions $\mathcal{S} = \{S_1, ..., S_n\}$ such that each solution S_i in $\mathcal{S}$ is a set of mutually consistent actions such that:

- each agent executes one action in S_i;
- the action set S_i is consistent with the current state (i.e., $g, p, pre(S_i) \not\vdash \bot$);
- the final state induced by the action set S_i is consistent (i.e., $g, post(S_i) \not\vdash \bot$).

The NM does not require that the preconditions of actions in a solution are met in the current state, since it is possible that the preconditions are true, but the NM is unaware of it.

Once all solutions are found, the NM calculates the prediction collection as follows:

$$\mathcal{P} = \{\mathcal{P}_\alpha : \forall \alpha \in G\}$$

where actions predicted for agent $\alpha \in G$ are defined as:

$$\mathcal{P}_\alpha = \{a : a \in S_i \wedge S_i \in \mathcal{S} \wedge actor(a) = \alpha\}$$

The problem with full prediction is that finding all the sequences of mutually consistent actions is *exponential*, as it requires to recursively search for all the sequences of mutually consistent actions that may be executed by agents. In the worst case, the temporal cost of this search is $O(|G|^{|D| \times I_D})$, where G is the set of agents, D is the set of action descriptions and I_D is the maximum number of instantiations per action (i.e., the number of ways in which variables in an action precondition can be instantiated). This situation arises when all actions are applicable for all agents. Thus, full prediction is only feasible for small scale scenarios.

Approximate Action Prediction. Approximate prediction performs an approximate search for the actions performed by agents that are consistent with the current state, but the NM relaxes the condition that actions are mutually consistent.

Definition 6. Given a partial state p (current state); we define *approximate search* as a function that calculates the approximate solution set $\widetilde{S} = \{a_i, ..., a_k\}$ such that:

- the preconditions of each action in $\widetilde{S}$ are consistent with the current state (i.e., $\forall a \in \widetilde{S} : g, p, pre(a) \not\vdash \bot$).

Once all solutions are found, the NM calculates the prediction collection as follows:

$$\mathcal{P} = \{\mathcal{P}_\alpha : \alpha \in G\}$$

where: $$\mathcal{P}_\alpha = \{a : a \in \widetilde{S} \wedge actor(a) = \alpha\}$$

EXAMPLE 6. *Given the partial state p_0 (defined in Example 5), the approximate search predicts the actions of agents $u1, u2$ and $u3$. In particular, the NM infers that $u1$ will perform action $start(u1)$ —this action is the only one consistent with p_0. The NM infers that $u2$ can perform three different actions $upload(u2, f1), download(u2, f1)$ and $close(u2, f1)$ —these three actions are the only ones consistent with p_0. Finally, the NM infers that $u3$ can perform three different actions $end(u3), open(u3, f1)$ and $open(u3, f2)$ —these three actions are the only ones consistent with p_0. The approximate solution set for this problem is defined as:*

$$\widetilde{S} = \{start(u1), upload(u2, f1), download(u2, f1), \\ close(u2, f1), end(u3), open(u3, f1), open(u3, f2)\}$$

and the sets of predicted actions are:

$$\mathcal{P}_{u1} = \{start(u1)\}$$
$$\mathcal{P}_{u2} = \{upload(u2, f1), download(u2, f1), close(u2, f1)\}$$
$$\mathcal{P}_{u3} = \{end(u3), open(u3, f1), open(u3, f2)\}$$

Approximate prediction can be computed in polynomial time by a filter algorithm that searches for each agent the actions it can perform in a given state. The temporal cost of this algorithm is $O(|G| \times |D| \times I_D)$.

3.3 Selection of Agents to be Monitored

Once the actions of agents have been predicted, the NM should select which agents will be monitored in the next step. When the NM predicts that an agent is only able to perform one action, the NM can be sure about the action that will be performed by the agent and there is no need to observe it as it can be taken for granted. In this case, the agent is deleted from the prediction collection and the action is added to the set of observations. The rest of the agents are ranked according their interest from a norm monitoring point of view. In particular, o agents with the highest rank are selected to be monitored.

The rank of a particular agent is calculated considering the chances it violates or fulfils a norm[3]. Given an agent $\alpha \in G$, the rank function ($R : G \rightarrow [0,1]$) is calculated as a combination of two factors as follows:

$$R(\alpha) = \underbrace{CF(\alpha)}_{\text{Confidence Factor}} \times \underbrace{IF(\alpha)}_{\text{Interest Factor}}$$

The interest factor estimates the probability of the action executed by α being of interest to norm monitoring; i.e., the probability of this action being an action that may violate or fulfil a norm[4]. Recall that the NM has partial knowledge about the state of the world and, as a result, the NM cannot be sure about the norms that are active at a given moment. To represent this, the rank function also considers the confidence factor, which is related to how certain the NM is about the interest factor. The rank function is defined as a product to ensure that it is: 0 when any of the factors is 0 (e.g., when the agent is not interesting for norm monitoring), increasing wrt. both factors and continuous[5].

Definition 7. Given an agent $\alpha \in G$, a set of predicted actions for that agent $\mathcal{P}_\alpha$, a partial state description p (current state), and a set of norms N; the interest set for agent α is calculated as the set of predicted actions that could violate a prohibition or fulfil an obligation:

$$I_\alpha = \left\{ a \left| \begin{array}{l} a \in \mathcal{P}_\alpha \wedge \exists \langle deontic, condition, action \rangle \in N \wedge \\ \exists \sigma : ((g, p, \sigma \cdot condition \nvdash \perp) \wedge (\sigma \cdot action = a) \end{array} \right. \right\}$$

The *interest factor* is defined as the ratio of the number of actions in the interest set to the number of actions in the agent predicted set:

$$IF(\alpha) = \frac{|I_\alpha|}{|\mathcal{P}_\alpha|}$$

Definition 8. Given an agent $\alpha \in G$, a set of predicted actions for that agent $\mathcal{P}_\alpha$, a partial state description p (current state), and a set of norms N; the confidence set for agent α is calculated as the set of predicted actions that surely violate a prohibition or fulfil an obligation:

$$C_\alpha = \left\{ a \left| \begin{array}{l} a \in \mathcal{P}_\alpha \wedge \exists \langle deontic, condition, action \rangle \in N \wedge \\ \exists \sigma : ((g, p \vdash \sigma \cdot condition) \wedge (\sigma \cdot action = a) \end{array} \right. \right\}$$

[3]Note we do not require that the NM has previous knowledge about agents' behaviour and it assumes all agents violate norms with the same probability (e.g., Intrusion Detection Systems [Hu *et al.*, 2008] cannot know a priori if an IP is malicious as IPs may change dynamically). However, if the NM observes that a particular agent tends to violate (vs. fulfil) norms more than others, then the chances it violates (vs. fulfil) norms could be weighted with these observed tendencies.

[4]We assumed all norms are of equal importance. If this is not the case, then the chances each agent violates/fulfils a norm could be pondered with the norm importance.

[5]The aim of this paper is not to analyse combination operations for data fusion, as this has been done in [Bloch, 1996]. In this paper, we make use of a well-known combination operator with characteristics suitable for selecting monitored agents.

The *confidence factor* is defined as the ratio of the number of actions in the confidence set to the number of actions that can be executed by the agent:

$$CF(\alpha) = \frac{|C_\alpha|}{|\mathcal{P}_\alpha|}$$

Once the NM selects the agents to be monitored (denoted by T), then it observes their actions and calculates the list of predicted actions for unobserved agents as follows:

$$L = \bigcup_{\substack{\forall \mathcal{P}_\alpha \in \mathcal{P}: \\ \alpha \notin T}} \mathcal{P}_\alpha$$

EXAMPLE 7. *Considering the sets of predicted actions for all agents (which have been calculated in Example 6), the action of user $u1$ can be taken for granted. Thus, $\mathcal{P}_{u1}$ is deleted from $\mathcal{P}$. Then, the NM calculates the rank for users $u2$ and $u3$:*

$$\begin{array}{ll} I_{u2} = C_{u2} = \{\} & R(u2) = \frac{0}{3} \times \frac{0}{3} = 0 \\ I_{u3} = C_{u3} = \{open(u3, f1)\} & R(u3) = \frac{1}{3} \times \frac{1}{3} = 0.17 \end{array}$$

In this case, the interest and confidence factors are the same as the NM has complete knowledge of the initial state ($p_0 \equiv s_0$). Assuming that only one agent can be monitored (i.e., $o = 1$), the NM decides to observe actions of user $u3$ as it is the one with the highest rank. In this case the list of predictions for unobserved agents is defined as follows:

$$L = \{upload(u2, f1), download(u2, f1), close(u2, f1)\}$$

3.4 State Update

As the NM only observes a subset of the actions performed by agents, the NM updates its representation of the world (p_t) based on a partial sequence of observed actions. At time t the NM carries out a monitoring activity and observes some of the actions performed by agents (Act_t). These actions have evolved s_t into the next state s_{t+1}. If all actions have been observed ($|Act_t| = |G|$), then the next partial state p_{t+1} can be constructed by considering the effect of actions in Act_t on p_t and the dynamic properties on p_t that have not been modified by these actions. A different case arises when the NM observes a subset of the actions performed by the agents ($|Act_t| < |G|$). In this case, the NM cannot be sure about the effects of unobserved actions. Thus, the new partial state p_{t+1} is constructed by considering the postconditions of the observed actions (i.e., positive postconditions are positive literals in p_{t+1} and negative postconditions are negative literals in p_{t+1}), the dynamic properties that have not been modified by the observed and predicted actions, and that the rest of dynamic propositions are unknown. Partial states in the general case are defined as:

Definition 9. Given a partial state description p_t corresponding to time t, and a sequence of observed actions Act_t executed by agents at time t and the list of predicted actions for unobserved agents L; the new partial state p_{t+1} resulting from executing actions Act_t in p_t is obtained as follows:

$$p_{t+1} = \begin{cases} \mathit{eff}(Act_t) \bigcup inv(p_t, Act_t) & \text{if } |Act_t| = |G| \\ post(Act_t) \bigcup inv(p_t, Act_t \cup L) & \text{otherwise} \end{cases}$$

where eff is the set formed by the *effects* of a set of actions; i.e., the effects are the postconditions of actions and the preconditions not invalidated by these postconditions. More formally, given a set of actions $A = \{a_1, ..., a_n\}$ its effects is a set of grounded literals as follows:

$$\mathit{eff}(A) = \left(\bigcup_{\substack{\forall pre \in pre(A): \\ pre, post(A) \nvdash \perp}} pre \right) \bigcup \left(\bigcup_{\forall post \in post(A)} post \right)$$

and inv is the set formed by *invariant* literals; i.e., literals of p_t that have not been modified by actions. More formally, given a partial state p and a set of actions $A = \{a_1, ..., a_n\}$, the invariant literals are defined as follows:

$$inv(p, A) = \bigcup_{\substack{\forall l \in p: \\ l, post(A) \nvdash \perp}} l$$

Besides that, the actions observed can also be used to increase the knowledge about the current state. In particular, the current state p_t is updated considering the preconditions of observed actions Act_t as follows:

$$p_t = p_t \bigcup pre(Act_t)$$

EXAMPLE 8. *The NM decides to observe action of user $u3$ and it can take for granted the action executed by $u1$ (i.e., $Act_0 = \{start(u1), open(u3, f1)\}$). The NM infers the dynamic propositions that are known in p_1 as follows:*

$$p_1 = post(Act_0) \bigcup inv(p_0, Act_0 \cup L)$$

where:

$$post(Act_0) = \{session(u1), opened(u3, f1)\}$$

$$inv(p_0, Act_0 \cup L) = \{session(u2), session(u3), \neg opened(u1, f1), \neg opened(u1, f2), \neg opened(u2, f2), \neg opened(u3, f2)\}$$

State p_0 remains unaltered as it was already complete.

3.5 Norm Monitoring

Once all the information about the actions performed by the agents has been analysed, the NM checks which instances have been violated or fulfilled. Given that the NM has partial knowledge about the current state of the world, the NM should control norms only when it is completely sure that the norms are relevant to ensure that the norm monitoring process is sound (e.g., the NM cannot indicate that a violation has occurred when it has not in fact occurred). In particular, we define that a norm is relevant to a partial situation when the norm condition is satisfied by the partial situation —i.e., a norm $\langle deontic, condition, action \rangle$ is relevant to a partial situation represented by a partial state p, and the static properties g; if $\exists \sigma$ such that $p, g \vdash \sigma \cdot condition$.

EXAMPLE 9. *In state p_0 the norm that forbids users to open files already opened is relevant and instantiated:*

$$\langle \mathcal{F}, open(U1, f1) \rangle \text{ where } \sigma = \{F/f1, U2/u2\}$$

Once the NM has determined the instances that are relevant, it checks compliance with these instances. If the NM has partial knowledge about the actions performed by agents, it can only determine that an obligation (vs. prohibition) instance has been fulfilled (vs. violated). If the NM knows all the actions performed by agents, it can determine whether an obligation or prohibition has been fulfilled or violated.

Definition 10. Given an norm instance $\langle D, action' \rangle$ and a set of actions Act, the instance is defined as:

$$\begin{cases} fulfilled & \text{iff } (D = \mathcal{O} \wedge \exists \sigma : \sigma \cdot action' \in Act) \text{ or} \\ & (D = \mathcal{F} \wedge \nexists \sigma : \sigma \cdot action' \in Act \wedge |Act| = |G|) \\ violated & \text{iff } (D = \mathcal{F} \wedge \exists \sigma : \sigma \cdot action' \in Act) \text{ or} \\ & (D = \mathcal{O} \wedge \nexists \sigma : \sigma \cdot action' \in Act \wedge |Act| = |G|) \\ unknown & \text{otherwise} \end{cases}$$

EXAMPLE 10. *Given that $Act_0 = \{start(u1), open(u3, f1)\}$, the NM detects that $u3$ has violated the prohibition norm instantiated in p_0; i.e., it opened file f1 while it was opened by $u2$.*

4 Evaluation

This section compares experimentally a NM with full prediction and a NM with approximate prediction to a *traditional* norm monitor. None of the existing norm monitoring approaches selects the agents to be monitored and they only monitor what they can see by chance. Therefore, in order to be able to compare our proposal with a traditional one, we model a traditional monitor as selecting agents randomly. Also, note that, due to the lack of space, we do not include the execution time as the cost of full or approximate prediction (exponential and polynomial, respectively) will always have to be added on top of any cost incurred by a traditional monitor, so we focus on whether this cost would actually be worth the increase in the detection of violations/fulfilments.

Extensive Simulation. We implemented a simulator in Java in which there is a set of agents that perform actions in a monitored environment described below. We conducted experiments in which the number of agents G took a random value within the $[\![1, 100]\!]$ interval. Besides that, to be able to compare with the full NM, we also considered small scenarios only, in which the number of agents G took a random value within the $[\![1, 5]\!]$, as the full prediction has an exponential cost and it is intractable for most of the cases with the default intervals. The number of actions A took a random value within the $[\![1, 20]\!]$ interval. The simulation is executed 100 steps.

We modelled different agent types with different capabilities to perform actions. To model these capabilities, a set of roles is created at the beginning of each simulation. The number of roles created took a random value within the $[\![1, A]\!]$ interval. For each role, a subset of actions are randomly selected as capabilities. To avoid that all roles have similar capabilities, the number of actions selected as role capabilities took a random value within the $[\![1, \lceil 0.1 * A \rceil]\!]$ interval (i.e., at maximum each role is capable of performing 10% of the actions). Each agent is defined as enacting a random subset of the roles. In each step of the simulation, each agent selects randomly one action to execute.

The environment is described in terms of different properties (grounded propositions) that can be true or false. The number of propositions P took a random value within the $[\![A, 2*A]\!]$ interval (i.e., there is at least one proposition per each action). Actions allow agents to change the state of the environment. At the beginning of each simulation, a set of actions is randomly generated. Besides these actions, a NOP action, which has no effect on the environment, was created. Agents' actions are regulated by a randomly-created set of norms. In particular the number of norms took a random value within the $[\![1, A]\!]$ (i.e., there is at maximum one norm per each action). To avoid that norms (actions) have too many constraints, which would be unrealistic and make norms (actions) to be only instantiated (executed) on very few situations, the number of propositions in the conditions took a random value within the $[\![1, \lceil 0.1*P \rceil]\!]$ interval.

To analyse the performance of monitors w.r.t. their capabilities to observe actions, we varied the ratio of observed actions, which is the number of agents that can be observed divided by the total ($o/|G|$). Table 1 shows the 99% confidence intervals for the percentage of detected violations[6]. The approximate NM offers on average a 57% performance improvement over a traditional monitor under partial observability conditions. A Kruskal-Wallis test also confirmed that there is a significant difference between the violations detected by the traditional monitor and the approximate NM ($\alpha = 0.01$). When compared to full NM in small scenarios, the full NM offers on average a 11% performance improvement over an approximate NM. The performance of traditional and approximate monitors is slightly worse in small scenarios because the number of agents is low w.r.t. the number of actions.

Obs. Ratio	Traditional Monitor	Approx. NM	Obs. Ratio	Traditional Monitor	Full NM	Approx. NM
0%	0$\pm$ 0%	0$\pm$0%	0%	0$\pm$0%	0$\pm$0%	0$\pm$0%
20%	13$\pm$1%	26$\pm$3%	20%	2$\pm$1%	22$\pm$3%	18$\pm$3%
40%	26$\pm$1%	43$\pm$3%	40%	5$\pm$1%	30$\pm$3%	26$\pm$3%
60%	40$\pm$2%	61$\pm$3%	60%	25$\pm$3%	52$\pm$4%	48$\pm$4%
80%	52$\pm$3%	76$\pm$4%	80%	47$\pm$4%	73$\pm$3%	68$\pm$3%
100%	100$\pm$0%	100$\pm$0%	100%	100$\pm$0%	100$\pm$0%	100$\pm$0%
$G \in [\![1, 100]\!]$ and $A \in [\![1, 20]\!]$			$G \in [\![1, 5]\!]$ and $A \in [\![1, 20]\!]$			

Table 1: Observability Experiment

Case Study. We considered a real data set of patient confidentiality laws in the US, which are state-specific laws that forbid health departments to release personally identifiable information for specific causes when patients have communicable diseases. Confidentiality laws in the US[7] cover 51 states with an average of over 57 regulations per state (for a total over 2900 laws). Doctors receive requests to *release* patients' data to other health departments and institutions for different *causes* (e.g., research). Depending on the state, the cause, and patient's diseases, the release of personally identifiable information without patient consent may be illegal. To comply with state laws, doctors should verify if a patient is affected by a law and, if need be, manually anonymise the patient textual data before sending it. Norm monitoring in this domain is extremely challenging, as health records are mostly *free text* written by doctors [Meystre *et al.*, 2008] and it is infeasible to have human operators investigating all data exchanges, so there will be incomplete observations for monitoring compliance with patient confidentiality laws.

We implemented this case study in Java so that compliance with the state norms is controlled by an NM. In each simulation, we randomly generate doctors $[\![1, 100]\!]$ based on each state, randomly assign $[\![1, 100]\!]$ patients to each doctor, and patients are assigned randomly some of 235 distinct diseases [Lozano *et al.*, 2013]. In each step of the simulation, requests of information are generated and each doctor chooses randomly to verify compliance with the confidentiality laws stated above or to send the data straight-away[8]. Table 2 shows the 99% confidence intervals for the percentage of detected violations per observation ratio. Due to the size of the problem, we could not get results for the full prediction in reasonable time. The approximate NM offers on average a 32% performance improvement over a traditional monitor under partial observability conditions.

Observation Ratio	Traditional Monitor	Approximate NM
0%	0 $\pm$0%	0$\pm$0%
20%	20$\pm\epsilon$%	33 $\pm\epsilon$%
40%	40$\pm\epsilon$%	57$\pm$2%
60%	60$\pm\epsilon$%	77$\pm$2%
80%	80$\pm\epsilon$%	95$\pm\epsilon$%
100%	100$\pm$0%	100$\pm$0%

Table 2: Patient Confidentiality Laws Case Study

5 Discussion

Most of the existing proposals on norm monitoring [Gaertner *et al.*, 2007; Minsky and Ungureanu, 2000; Modgil *et al.*, 2009] assume that monitors have complete observations. Exception to these approaches are two recent proposals [Bulling *et al.*, 2013; Alechina *et al.*, 2014]. In [Bulling *et al.*, 2013], the partial observability problem is addressed by combining different norm monitors to build ideal monitors (i.e., monitors that together are able to monitor actions of all agents). In [Alechina *et al.*, 2014], the authors propose to synthesise an approximate set of norms that can be monitored given the observation capabilities of a monitor. However, there are circumstances in which norms cannot be modified (e.g., contract and law monitoring) or ideal monitors are expensive and/or not feasible. We take a different approach in which norms and monitors' observation capabilities remain unchanged and monitors select which agents are monitored based on two different prediction processes: full and approximate. This obviously adds a cost to traditional norm monitoring. However, our experiments demonstrate that a NM selecting agents to be monitored using any of these prediction processes detects significantly more violations than a traditional monitor, and in particular, it shows the polynomial cost that approximate would introduce is well worth the 32%-57% gain on detected violations as shown in the evaluations, which would be missed by a traditional monitor.

[6] Similar results are obtained in case of fulfilments.

[7] `http://lawatlas.org/query?dataset=public-healthdepartments-and-state-patient-confidentiality-laws`

[8] We sought to study the performance of different norm monitors *ceteris paribus* (i.e., without the noise introduced by specific doctor behaviour, law enforcement, etc.).

References

[Alechina *et al.*, 2014] Natasha Alechina, Mehdi Dastani, and Brian Logan. Norm approximation for imperfect monitors. In *Proc. of AAMAS*, pages 117–124, 2014.

[Bloch, 1996] Isabelle Bloch. Information combination operators for data fusion: a comparative review with classification. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 26(1):52–67, 1996.

[Boutilier and Brafman, 2001] Craig Boutilier and Ronen I Brafman. Partial-order planning with concurrent interacting actions. *Journal of Artificial Intelligence Research*, 14(1):105–136, 2001.

[Bulling *et al.*, 2013] Nils Bulling, Mehdi Dastani, and Max Knobbout. Monitoring norm violations in multi-agent systems. In *Proc. of AAMAS*, pages 491–498, 2013.

[Fitting, 1996] Melvin Fitting. *First-order logic and automated theorem proving*. Springer, 1996.

[Gaertner *et al.*, 2007] Dorian Gaertner, Andres Garcia-Camino, Pablo Noriega, J-A Rodriguez-Aguilar, and Wamberto Vasconcelos. Distributed norm management in regulated multiagent systems. In *Proc. of AAMAS*, pages 624–631, 2007.

[Hu *et al.*, 2008] Weiming Hu, Wei Hu, and Steve Maybank. Adaboost-based algorithm for network intrusion detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 38(2):577–583, 2008.

[López y López *et al.*, 2006] Fabiola López y López, Michael Luck, and Mark d'Inverno. A normative framework for agent-based systems. *Computational & Mathematical Organization Theory*, 12(2-3):227–250, 2006.

[Lozano *et al.*, 2013] Rafael Lozano, Mohsen Naghavi, Kyle Foreman, Stephen Lim, Kenji Shibuya, Victor Aboyans, Jerry Abraham, Timothy Adair, Rakesh Aggarwal, Stephanie Y Ahn, et al. Global and regional mortality from 235 causes of death for 20 age groups in 1990 and 2010: a systematic analysis for the global burden of disease study 2010. *The Lancet*, 380(9859):2095–2128, 2013.

[Meystre *et al.*, 2008] Stéphane M Meystre, Guergana K Savova, Karin C Kipper-Schuler, John F Hurdle, et al. Extracting information from textual documents in the electronic health record: a review of recent research. *Yearb Med Inform*, 35:128–44, 2008.

[Minsky and Ungureanu, 2000] N.H. Minsky and V. Ungureanu. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *ACM Transactions on Software Engineering and Methodology*, 9(3):273–305, 2000.

[Modgil *et al.*, 2009] S. Modgil, N. Faci, F. Meneguzzi, N. Oren, S. Miles, and M. Luck. A framework for monitoring agent-based normative systems. In *Proc. of AAMAS*, pages 153–160, 2009.

[Vasconcelos *et al.*, 2007] Wamberto Vasconcelos, Martin J Kollingbaum, and Timothy J Norman. Resolving conflict and inconsistency in norm-regulated virtual organizations. In *Proc. of AAMAS*, pages 632–639. ACM, 2007.