Norm Monitoring under Partial Action Observability

Natalia Criado and Jose M. Such, *Member, IEEE*

***Abstract*—In the context of using norms for controlling multi-agent systems, a vitally important question that has not yet been addressed in the literature is the development of mechanisms for monitoring norm compliance under partial action observability. This paper proposes the reconstruction of unobserved actions to tackle this problem. In particular, we formalise the problem of reconstructing unobserved actions, and propose an information model and algorithms for monitoring norms under partial action observability using two different processes for reconstructing unobserved actions. Our evaluation shows that reconstructing unobserved actions increases significantly the number of norm violations and fulfilments detected.**

***Index Terms*—Norm Monitoring, Action Observability.**

I. Introduction

Within the Multi-agent System (MAS) area, norms are understood as means to coordinate and regulate the activity of autonomous agents interacting in a given social context [19]. The existence of autonomous agents that are capable of violating norms entails the development of norm control mechanisms that implement norms in agent societies.

In the existing literature, several authors have proposed infrastructures to observe agent actions and detect norm violations upon them [1], [21]. The majority of these proposals have focused on providing efficient and scalable methods to monitor norms in dynamic agent societies, but they assume that all actions of agents are observable. However, this assumption is too strong because it is not necessarily true that all actions to be controlled can always be observed. One reason for this is that observing actions usually entails high costs. For example, the costs of setting, maintaining, and managing traffics radars to detect car speeds are very high, so traffic authorities usually decide to install a few of them in specific and critical locations. Another reason is that illegal actions may take place outside the institution controlled by the monitor; however, the effects of these actions can still be detected within the institution. For example, black market transactions cannot be directly observed by legal authorities, yet the corresponding money laundering transactions can be detected and sanctioned by these authorities.

Very recent work on norm monitoring under partial action observability proposes solutions to ensure complete action observability by increasing the actions that are observed, either by adding more monitors [5] or by adapting the norms to what can be observed [2]. However, these solutions are not always appropriate or feasible. For instance, in e-markets, such as eBAY[1] or Amazon[2], it is not possible to change trading laws to what can be observed. This paper goes beyond these approaches by also considering actions that were not observed but that can be reconstructed from what was observed.

The main contributions of this paper are: (i) a formalisation of the problem of reconstructing unobserved actions from observed actions for the purpose of norm monitoring; (ii) an exhaustive and an approximation solution to this problem; and (iii) an information model and algorithms used to monitor norms under partial action observability. Through an extensive empirical evaluation, we show that reconstructing unobserved actions increases noticeably the number of norm violations and fulfilments detected.

This paper is organised as follows: Section II contains the preliminary definitions used in this paper. Section III describes the information model of norm monitor proposed in this paper. Section IV contains the algorithms executed by norm monitors. Our proposal is evaluated in SectionV. Related word is discussed in Section VI. Finally, conclusions are contained in Section VII.

II. 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 ($\top$) and false propositions ($\perp$), and an infinite set of variables. The predicate and constant symbols are written as any sequence of alphanumeric characters beginning with a lower case letter. Variables are written as any sequence of alphanumeric characters beginning with a capital letter. We also assume the standard notion of substitution of variables [13]; 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 and objects that characterise the properties of the 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[3] 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 assumption" and defined by a set of static properties g and a state s. Moreover, there is a set of inference rules ($\bigtriangledown$) representing domain knowledge.

Example. *In this paper we will use a running example in which there are three robots that should attend requests at six offices in a building. The goal of the robots is to attend these requests as soon as possible. Figure 1a depicts our initial*

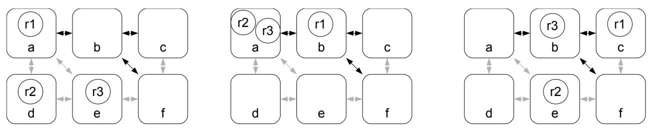
N. Criado is with the School of Computing and Mathematical Sciences, Liverpool John Moores University, UK, e-mail: n.criado@ljmu.ac.uk.

J. M. Such is with the School of Computing and Communications Infolab21, Lancaster University, UK, email:j.such@lancaster.ac.uk.

[1]http://www.ebay.com

[2]http://www.amazon.com

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



(a) Initial State s_0 (b) State s_1 (c) State s_2

Fig. 1: Example Scenario. Offices are represented by squares, agents are represented by circles and the corridors are represented by arrows. Black arrows correspond to corridors observed by the Norm Monitor (NM) and grey arrows correspond to corridors not observed by the NM.

scenario. In our example, the language $\mathcal{L}$ contains: 4 predicate symbols ($robot, office, in, corridor$), used to represent the robots and offices, the positions of the robots and the connections between offices in the building; 3 constant symbols to represent the robots ($r1, r2, r3$); and 6 constant symbols to represent the offices (a, b, c, d, e, f). The information about the robots, offices and corridors between offices is static and represented as follows:

$$g = \{robot(r1), robot(r2), robot(r3), office(a), ..., office(f), \\ corridor(a,b), corridor(b,a), ..., corridor(e,a)\}$$

The information about the location of the robots is dynamic. Specifically, the initial state s_0 is defined as follows:

$$s_0 = \{in(r1,a), in(r2,d), in(r3,e)\}$$

In this domain there is an inference rule ($\bigtriangledown$) representing that a robot cannot be in two different offices at the same time:

$$\bigtriangledown = \{\{in(R1,OA), in(R1,OB), OA \neq OB\} \vdash \bot\}$$

A. Action Definitions

D is a finite set of action descriptions that induce state transitions. An action description d is represented using preconditions and postconditions. If a situation does not satisfy the preconditions, then the action cannot be applied in this situation. 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 literals appearing in the postconditions are deleted and all positive literals in the postconditions are added. Moreover, actions are executed in a MAS and, as a consequence, we need to be able to represent concurrent actions with interacting effects. For the sake of simplicity, we will represent concurrent actions without an explicit representation of time[4] as proposed in [4]. The main idea beyond this representation is that individual agent actions do interact (i.e., one action might only achieve the intended effect if another action is executed concurrently). Specifically, each action is also represented by a (possibly empty) concurrent condition that describes the actions that must (or cannot) be executed concurrently[5].

[4]An explicit representation of time may play a role on other problems like scheduling concurrent actions, but is not strictly necessary for monitoring the effects of interaction.

Definition 1. *An action description d is a tuple $\langle name, pre, con, post\rangle$ 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;*
- *con is the concurrent condition; i.e., a set of positive and negative action schemata[6], some of which can be partially instantiated or constrained;*
- *$post$ 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 $pre(d), con(d), post(d)$ the action precondition, concurrent condition and postcondition.

EXAMPLE. *In our example, there is only one action that can be executed by robots:*

$$\left\langle \begin{array}{l} move, \{robot(R), office(O1), office(O2), in(R,O1), \\ corridor(O1,O2)\}, \{\}, \{\neg in(R,O1), in(R,O2)\} \end{array} \right\rangle$$

This action represents the movement of a robot from one office to another. The parameters of this action are the robot (R), the source office ($O1$), the destination office ($O2$). To execute this action, the robot should be located at the source office and the two offices should be connected. Once the operation has been applied, the robot is no longer at the source office and it is at the destination office.

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

- *There is a substitution σ of variables in pre, such that the precondition is satisfied (i.e., entalied by) the situation; i.e., $s, g \vdash \sigma \cdot pre$;*
- *$\sigma \cdot pre, \sigma \cdot post$ are grounded;*
- *pre' is a set of grounded literals in $\sigma \cdot pre$ containing dynamic properties only;*
- *$post' = \sigma \cdot post$ and $con' = \sigma \cdot con$.*

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

EXAMPLE. *In state s_0, the robot $r1$ moves from office a to office b. This is formalised as follows:*

$$\left\langle \begin{array}{l} move, \{robot(r1), office(a), office(b), in(r1,a), \\ corridor(a,b)\}, \{\}, \{\neg in(r1,a), in(r1,b)\} \end{array} \right\rangle$$

[5]A more sophisticated definition of the concurrent condition would allow actions to have conditional effects according to the actions that are executed concurrently. Without loss of expressiveness, we will not consider conditional effects in action descriptions (note that any action with conditional effects can be represented by a set of actions with non conditional effects).

[6]An action schema contains an action name and the parameters of this action. Note that positive action schemata are implicitly existentially quantified –i.e., one instance of each positive schema must occur concurrently– and negative schemata are implicitly universally quantified.

In a MAS, concurrent actions[7] define state transitions. More formally, a concurrent action $A = \{a_1, ..., a_n\}$ is a set of individual actions. 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 $actor(A) = \bigcup actor(a_i)$.

Given a concurrent action $A = \{a_1, ..., a_n\}$ we say that the concurrent condition of an individual action a_i of A is *satisfied* when for all positive schema in the concurrent condition exists an action a_j $(i \neq j)$ in A, such that a_j is an instance of the schema; and for all negative schema none of the elements in A is an instance of the schema. For the sake of simplicity, we assume that each agent performs one action at a time[8].

Definition 3. *(Consistency [4]) Given a concurrent action $A = \{a_1, ..., a_n\}$ it is consistent if:*

- *$pre(A)$ is consistent (i.e, $pre(A) \not\vdash \perp$);*
- *$post(A)$ is consistent (i.e, $post(A) \not\vdash \perp$);*
- *the concurrent condition of each action is satisfied;*
- *the concurrent action is complete (i.e., each agent performs one action in A).*

EXAMPLE. *The concurrent action $A = \{move(r1, a, b), move(r2, d, a), move(r3, e, a)\}$[9] is consistent since:*

- *$pre(A) = \{in(r1, a), in(r2, d), in(r3, e)\}$ which is consistent;*
- *$post(A) = \{in(r1, b), in(r2, a), in(r3, a), \neg in(r1, a), \neg in(r2, d), \neg in(r3, e)\}$ which is consistent;*
- *the concurrent conditions of both actions are satisfied;*
- *each robot performs one action.*

A concurrent action $A = \{a_1, ..., a_n\}$ is applicable in a situation if A is consistent and each individual action $a_i \in A$ is applicable in this situation.

Given a consistent action, we define its effects as the postconditions of its individual actions and the preconditions not invalidated by the postconditions. More formally, given a concurrent action $A = \{a_1, ..., a_n\}$ its effects are a set of grounded literals as follows:

$$eff(A) = (\bigcup_{\substack{\forall pre \in pre(A): \\ pre, post(A) \not\vdash \perp}} pre) \bigcup (\bigcup_{\forall post \in post(A)} post)$$

B. Norm Definitions

We consider *norms* as formal statements that define patterns of behaviour by means of *deontic modalities* (i.e., *obligations* and *prohibitions*). Specifically, our proposal is based on the notion of norm as a conditional rule of behaviour that defines under which circumstances a pattern of behaviour becomes relevant and must be fulfilled [21], [29], [19], [11].

Definition 4. *A norm is defined as a tuple $\langle deontic, condition, action \rangle$, where:*

- *$deontic \in \{\mathcal{O}, \mathcal{P}\}$ is the deontic modality of the norm, determining if the norm is an obligation ($\mathcal{O}$) or prohibition ($\mathcal{P}$);*
- *$condition$ is a set of literals of $\mathcal{L}$ as well as equality and inequality constraints that represents the norm condition, i.e., it denotes the situations in which the norm is relevant.*
- *$action$ is a positive action schema that represents the action controlled by the norm.*

EXAMPLE. *In our example, there is a norm that avoids collisions by forbidding any robot to move into an office when the office is occupied by another robot:*

$$\langle \mathcal{P}, in(R1, L2), move(R2, L1, L2) \rangle$$

This norm states that when a robot $R1$ is located in office $O2$ other robots are forbidden to move from any office $L1$ to $L2$.

In line with related literature [1], [20], [2], 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 also define that a norm is relevant to a specific situation if the norm condition is satisfied in the situation. Besides, we define that a norm condition is satisfied in a given situation when 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.

Definition 5. *Given a specific situation denoted by a state s and a set of static properties g, and a norm $\langle deontic, condition, action \rangle$; a norm instance is a tuple $\langle deontic, action' \rangle$ such as:*

- *There is a substitution σ such that the condition is satisfied in the situation; i.e., $s, g \vdash \sigma \cdot condition$;*
- *$action' = \sigma \cdot action$.*

EXAMPLE. *In state s_0 the norm that forbids robots to move into occupied offices is instantiated as follows:*

$$\langle \mathcal{P}, move(R2, L1, d) \rangle \text{ where } \sigma = \{L2/d\}$$
$$\langle \mathcal{P}, move(R2, L1, a) \rangle \text{ where } \sigma = \{L2/a\}$$
$$\langle \mathcal{P}, move(R2, L1, e) \rangle \text{ where } \sigma = \{L2/e\}$$

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. We classify detected violations (vs. fulfilments) into: *identified* violations (vs. fulfilment), which refers to when the monitor knows the specific action that an agent executed and violates (vs. fulfils) an instance; and *discovered* violations (vs. fulfilment), which refers to when the monitor knows that an agent violated (vs. fulfilment) some instance but does not know the forbidden (vs. mandatory) action executed by the agent.

III. NM INFORMATION MODEL

Let us assume a set of agents Ag to be monitored, a set of norms N that regulate the actions of agents, and a set D of action descriptions that represent the actions that can be performed by agents. For the sake of simplicity, we assume that there is a single Norm Monitor (NM) that observes the

[7]Concurrent action means actions that occur at the same time and does not necessarily imply agent cooperation or coordination.

[8]This limitation can be relaxed by decomposing agents into groups of agents corresponding to agents' actuators [4].

[9]For simplicity, we represent actions by their schemata.

actions performed by agents and monitors norm compliance[10]. We also assume that actions are deterministic and that the current state evolves due to action execution only[11]. The goal of the NM is to analyse a partial sequence of action observations to detect norm violations. The enforcement of norms 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).

A. State Representation

As the NM only observes 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.

To begin with, assume that the NM monitor has complete knowledge of the initial state (this will be relaxed later). Thus, at t_0 the NM knows which grounded atomic formulas are true or false in the initial state ($p_0 \equiv s_0$). From that moment on, the NM monitors the actions performed by agents at each point in time. At time t_0 the NM carries out a monitoring activity and observes some of the actions performed by agents (Act_0). These actions have evolved s_0 into a new state s_1. As previously mentioned, the NM has limited capabilities for observing the actions performed by agents. Thus, it is possible that the NM observes a subset of the actions performed by agents. Specifically, if all actions have been observed ($|Act_0| = |Ag|$), then the resulting partial state p_1 can be constructed by considering the effects of actions in Act_0 on p_0 so $p_1 \equiv s_1$. A different case arises when the NM observes a subset of the actions performed by the agents ($|Act_0| < |Ag|$). In this case, the agent cannot be sure about the effects of unobserved actions. Thus, the new partial state p_1 is constructed by assuming that the postconditions of the observed actions must hold on state s_1 (i.e., positive postconditions are positive literals in p_1 and negative postconditions are negative literals in p_1) and the rest of dynamic propositions are unknown. If the NM takes into account the next sequence of actions that it observes at time t_1 (Act_1), then the NM can also infer that the preconditions of these actions must hold on state s_1, and, as a consequence, new propositions can be taken for sure in the partial state p_1, retrospectively. Partial states in the general case are defined as:

Definition 6. *Given a partial state description p_t corresponding to time t, and two consecutive sequences of observed actions Act_t and Act_{t+1} executed by agents at times t and $t+1$, respectively; the new partial state p_{t+1} resulting from executing actions Act_t in p_t and actions Act_{t+1} in p_{t+1} is obtained as follows:*

$$p_{t+1} = \begin{cases} post(Act_t) \bigcup pre(Act_{t+1}) & \textit{if } |Act_t| < |Ag| \\ p_t^* \bigcup \mathit{eff}(Act_t) \bigcup pre(Act_{t+1}) & \textit{otherwise} \end{cases}$$

where p_t^* is the set of invariant literals; i.e., literals of p_t that have not been modified by the actions in Act_t and it is defined as follows:

$$p_t^* = \bigcup_{\substack{\forall l \in p_t: \\ l, \mathit{eff}(Act_t) \not\vdash \perp}} l$$

EXAMPLE. *In our example, the NM knows which grounded atomic formulas are true or false in the initial state:*

$$\begin{aligned} p_0 = \{&in(r1,a), \neg in(r1,b), \neg in(r1,c), \neg in(r1,d), \neg in(r1,e), \\ &\neg in(r1,f), in(r2,d), \neg in(r2,a), \neg in(r2,b), \neg in(r2,c), \\ &\neg in(r2,e), \neg in(r2,f), in(r3,e), \neg in(r3,a), \neg in(r3,b), \\ &\neg in(r3,c), \neg in(r3,d), \neg in(r3,f)\} \end{aligned}$$

The NM has some surveillance cameras to monitor the movement of robots in the building. Specifically, the corridors that are monitored are the ones between offices: a and b; b and c; and b and f. These corridors are represented by black arrows in Figure 1, whereas non-monitored corridors are represented by grey arrows. In the initial state (s_0) depicted in Figure 1a, the robots execute the actions $move(r1,a,b), move(r2,d,a)$ and $move(r3,e,a)$ resulting in a new state (s_1) depicted in Figure 1b. However, the NM only observes the action of robot $r1$, because this action takes place in a monitored corridor; i.e., $Act_0 = \{move(r1,a,b)\}$. In the next state s_1, the robots execute actions $move(r1,b,c), move(r2,a,e)$ and $move(r3,a,b)$ resulting in a new state (s_2) depicted in Figure 1c. In this case the NM observes two actions; i.e., $Act_1 = \{move(r1,b,c), move(r3,a,b)\}$. Considering these two sets of observed actions the NM is able to infer the dynamic propositions that are known in s_1 as follows:

$$p_1 = \{in(r1,b), \neg in(r1,a), in(r3,a)\}$$

If the NM uses the information about the states and the observed actions, then no violation of the norm is detected and no robot is sanctioned. However, $r2$ and $r3$ have violated the norm, since they have moved into an occupied office through non-monitored corridors.

B. Action Reconstruction

NMs use Definition 6 to generate partial state descriptions based on the observed actions. Additionally, we propose that NMs reconstruct the actions that have not been observed. This reconstruction process entails: (i) searching for the actions that have been performed by unobserved agents; and (ii) using the actions found to increase the knowledge about the state of the world. The reconstruction process must be sound, e.g., it cannot indicate that a violation has occurred when it has not in fact occurred. In the following, we introduce full and approximate methods for reconstructing unobserved actions.

1) Full Reconstruction: Full reconstruction tries to find exhaustively the actions performed by all the agents that have not been observed. To this aim, the full reconstruction performs a search to identify all solutions to the reconstruction problem.

[10]However, our model can be used by a team of monitors as well.
[11]This assumption could be relaxed if NMs have capabilities for observing both state changes and actions.

Definition 7. *Given a partial state description p_t corresponding to time t (named initial state), a set of observed actions Act_t at time t, and an partial resulting state p_{t+1} corresponding to time $t+1$ (named final state); we define search as a function that computes sets of solutions $\mathcal{S} = \{S_1, ..., S_k\}$ such that each solution S_i in $\mathcal{S}$ is a set of actions such that:*

- *the concurrent action $S_i \cup Act_t$ is consistent;*
- *the initial state induced by the concurrent action $S_i \cup Act_t$ is consistent (i.e., $g, p_t, pre(S_i \cup Act_t), \nabla \not\vdash \bot$);*
- *the final state induced by the concurrent action $S_i \cup Act_t$ is consistent (i.e., $g, p_{t+1}, post(S_i \cup Act_t), \nabla \not\vdash \bot$).*

Thus, a solution is a set of actions performed by the agents that have not been observed[12] that are consistent with the states of the world before and after the execution of the actions. Given that the NM has a partial knowledge of the states, we do not require that the preconditions (vs. postconditions) of actions in a solution are met in the initial (vs. final) state, since it is possible that the preconditions (vs. postconditions) are true, but the NM is unaware of it.

EXAMPLE. *Given the partial state description p_0, the set of observed actions Act_0, and the partial resulting state p_1, the search function looks for actions of agents $r2$ and $r3$ (since they are the agents that have not been observed). According to the initial position of $r2$, the NM can infer that $r2$ may have performed two different actions $move(r2, d, a)$ and $move(r2, d, e)$ —these two actions are the only ones consistent with p_0. Similarly, the NM can infer that $r3$ may have performed three different actions $move(r3, e, a)$, $move(r3, e, d)$ and $move(r3, e, f)$ —these three actions are the only ones consistent with p_0. However, the actions $move(r3, e, d)$ and $move(r3, e, f)$ are not consistent with the final state —recall that these two actions have as postcondition the fact that $r3$ is in offices d and f, respectively; that p_1 defines that $r3$ is in office a; and that ∇ defines as inconsistent states where any robot is at two different locations. As a result, the solution set for this problem is defined as:*

$$\mathcal{S} = \{\{move(r2, d, a), move(r3, e, a)\},\\ \{move(r2, d, e), move(r3, e, a)\}\}$$

Once all solutions are found, the NM uses this information to extend the information about the actions performed by unobserved agents and the state of the world. To ensure that the reconstruction is sound, the NM calculates the intersection of actions in the solutions to select actions it is completely sure about (i.e., actions belonging to all solutions). Given a set of search solutions $\mathcal{S} = \{S_1, ..., S_k\}$ for some initial and final states, we define the *reconstruction* action set as follows:

$$R = \bigcap_{\forall S_i \in \mathcal{S}} S_i$$

If $R \neq \emptyset$, then the NM expands its knowledge about the actions performed by agents and it uses this information to increase the knowledge about the initial and final states. More formally, the set of actions observed in t is updated as:

$$Act_t = Act_t \cup R$$

[12]If all actions were observed, no reconstruction would be needed.

The initial state p_t is updated as follows:

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

Finally, the final state is updated as follows:

$$p_{t+1} = \begin{cases} p_{t+1} \bigcup post(R) \bigcup p_t^{\bullet} & \text{if } |Act_t| < |Ag| \\ p_{t+1} \bigcup p_t^{*} \bigcup eff(Act_t) & \text{otherwise} \end{cases}$$

where p_t^* is defined as before and $p_t^{\bullet}$ is the set of extended invariant literals; i.e., literals in p_t that have not been modified since there is not a solution $S_i \in \mathcal{S}$ such that the concurrent action $S_i \cup Act_t$ changes any of these literals:

$$p_t^{\bullet} = \bigcup_{\substack{\forall l \in p_t: \\ \nexists S_i \in \mathcal{S}: l, post(Act_t \cup S_i) \not\vdash \bot}} l$$

EXAMPLE. *The reconstruction set for the example is:*

$$R = \{move(r3, e, a)\}$$

This action belongs to all solutions, so the NM can be absolutely sure about the performance of this action, even when the NM has not observed it. As a consequence, the NM extends its information as follows:

$$Act_0 = \{move(r1, a, b), move(r3, e, a)\}$$

and p_0 remains unchanged and p_1 is updated as follows:

$$p_1 = \{in(r1, b), \neg in(r1, a), \neg in(r1, c), \neg in(r1, d), \neg in(r1, e),\\ \neg in(r1, f), \neg in(r2, b), \neg in(r2, c), \neg in(r2, f), in(r3, a),\\ \neg in(r3, b), \neg in(r3, c), \neg in(r3, d), \neg in(r3, e), \neg in(r3, f)\}$$

The main disadvantage of full reconstruction is that, for many real-world problems, the number of candidate solutions that needs to be explored is prohibitively large, as shown later in Section IV. In response to this problem, we provide a polynomial approximation below.

2) Approximate Reconstruction: Approximate reconstruction includes an approximate search that finds the actions performed by unobserved agents that are consistent with the states of the world before and after action execution. Specifically, approximate reconstruction identifies actions that do not necessarily include the specific actions performed by unobserved agents but that allow the NM to control norms. The main intuition beyond approximate reconstruction is as follows: imagine that at a given initial state an agent can perform just one action and that this action is forbidden (vs. mandatory). In this case, the NM *identifies* that the agent has violated (vs. fulfilled) a norm. Besides that, if an agent can perform n different actions and all these actions are forbidden (vs. mandatory), the NM does not need to know which action has been executed to conclude that a norm has been violated (vs. fulfilled)[13]. Hence, we say that a violation has been *discovered* (instead of *identified*). Given a set of

[13]Note that the propose of this paper is to monitor norms, not to determine whether agents are responsible for norm violations/fulfilments. Monitoring situations where agents can only execute forbidden/obligatory actions can help to detect norm-design problems. Additionally, the fact that an agent can only execute forbidden actions may be explained by the agent putting itself into these illegal situations (e.g., I am allowed to overtake but overtaking may put me in a situation where I can only exceed the speed limit).

prohibition instances P and an action a, we define that the action a is forbidden (denoted by $forbidden(P, a)$) when $\exists p \in P : \exists \sigma : \sigma \cdot action(p) = a$. Similarly, given a set of obligation instances O and an action a, we define that the action a is mandatory (denoted by $mandatory(O, a)$) when $\exists o \in O : \exists \sigma : \sigma \cdot action(o) = a$.

Definition 8. *Given a partial state p_t, a set of observed actions Act_t at time t, and a partial resulting state p_{t+1}; we define approximate search as a function that calculates the set of all unobserved applicable actions $\widetilde{S} = \{a_i, ..., a_n\}$ such that:*

- *the preconditions of each action in $\widetilde{S}$ are consistent with the initial state (i.e., $\forall a_i \in \widetilde{S} : g, p_t, pre(a_i), \nabla \not\vdash \perp$);*
- *the postconditions of each action in $\widetilde{S}$ are consistent with the final state (i.e., $\forall a_i \in \widetilde{S} : g, p_{t+1}, post(a_i), \nabla \not\vdash \perp$);*
- *actions in $\widetilde{S}$ are performed by unobserved agents (i.e., $actor(\widetilde{S}) \cap actor(Act_t) = \emptyset$);*
- *all unobserved agents perform at least one action in $\widetilde{S}$.*

Example. *Given the partial state description p_0, the set of observed actions Act_0, and the partial resulting state p_1, the approximate search function looks for actions of agents $r2$ and $r3$ (since they are the agents that have not been observed). According to the initial position of $r2$, the NM can infer that $r2$ may have performed two different actions $move(r2, d, a)$ and $move(r2, d, e)$. Again, $r3$ may have performed action $move(r3, e, a)$. The approximate solution for this problem is defined as:*

$$\widetilde{S} = \{move(r2, d, a), move(r2, d, e), move(r3, e, a)\}$$

As in full reconstruction, the NM uses approximate search solutions ($\widetilde{S}$) to expand its knowledge about the actions performed by unobserved agents and to increase the knowledge about the initial and final states. When an unobserved agent may have executed only one action, then the NM knows for sure that this action was executed. More formally, the *reconstruction* action set is defined as follows:

$$R = \bigcup_{\substack{\forall a \in \widetilde{S} : \nexists a' \in \widetilde{S}: \\ a \neq a' \wedge actor(a) = actor(a')}} a$$

The set of actions observed in t is updated as:

$$Act_t = Act_t \cup R$$

Then the initial state p_t is updated as follows:

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

The final state is updated as follows:

$$p_{t+1} = \begin{cases} p_{t+1} \bigcup post(R) \bigcup p_t^\circ & \text{if } |Act_t| < |Ag| \\ p_{t+1} \bigcup p_t^* \bigcup \mathit{eff}(Act_t) & \text{otherwise} \end{cases}$$

where p_t^* is defined as before and p_t° is the set of extended invariant literals in p_t; i.e., literals that have not been modified since there is not an observed action or an applicable action that changes them:

$$p_t^\circ = \left(\bigcup_{\substack{\forall l \in p_t: \\ l, post(Act_t) \not\vdash \perp}} l \right) \bigcap \left(\bigcup_{\substack{\forall l \in p_t: \\ l, post(\widetilde{S}) \not\vdash \perp}} l \right)$$

Finally, the set of discovered violations and fulfilments is a set of actions defined as follows:

$$D = \{a_1, ..., a_j\}$$

where for each action a_i in D: a_i is in $\widetilde{S}$ and the agent that performs a_i (i.e., $actor(a_i)$) is:

- able to execute more than one action (i.e., $\exists a_j \in \widetilde{S} : a_i \neq a_j \wedge actor(a_i) = actor(a_j)$);
- only able to execute forbidden (vs. mandatory) actions and a_i is one of these forbidden (vs. mandatory) actions;

When an agent is only able to perform forbidden (vs. mandatory) actions, an action among these can be selected according to various criteria. For example, in a normative system where the presumption of innocence principle holds, the NM should assume that the agent has violated (vs. fulfilled) the least (vs. most) important norm and the action that violates (vs. fulfils) this norm is selected. Note discovering violations is very useful in many practical applications, in which it would allow the NM to ban offender agents (e.g., Intrusion Detection/Prevention Systems [3]), to stop the execution of any offender agent (e.g., Business Process Compliance monitoring [27]), or to put offender agents under close surveillance (e.g., Model-Based Diagnosis Systems [22]), even when the specific action performed ins not known.

Example. *In case of the approximate reconstruction, $r3$ is only able to perform one action, which entails that the NM can be absolute sure about the performance of this action and the reconstruction set is defined as:*

$$R = \{move(r3, e, a)\}$$

As a consequence, the NM extends its information as follows:

$$Act_0 = \{move(r1, a, b), move(r3, e, a)\}$$

p_0 remains unchanged and p_1 is updated as follows:

$$\begin{aligned} p_1 = \{&in(r1, b), \neg in(r1, a), \neg in(r1, c), \neg in(r1, d), \neg in(r1, e), \\ &\neg in(r1, f), \neg in(r2, b), \neg in(r2, c), \neg in(r2, f), in(r3, a), \\ &\neg in(r3, b), \neg in(r3, c), \neg in(r3, d), \neg in(r3, e), \neg in(r3, f)\} \end{aligned}$$

In this situation, $r2$ is only able to execute forbidden actions —recall that the instances $\langle \mathcal{P}, move(R2, L1, a) \rangle$ and $\langle \mathcal{P}, move(R2, L1, e) \rangle$ forbid any robot to move into offices a and e and that $r2$ may have been executed actions $move(r2, d, a)$ and $move(r2, d, e)$. Thus, the set of discovered violations and fulfilments is defined as follows:

$$D = \{move(r2, d, e)\}$$

note that the discovered violation does not correspond to the action executed by $r2$, however, it allows the NM to determine that $r2$ must have violated an instance.

C. Norm Monitoring

Once all the information about the actions performed by the agents and the partial states has been reconstructed, the NM checks the actions of agents to determine which instances have been violated or fulfilled. Recall that norms in our model are defined as conditional rules that state which actions are obligatory or forbidden. Given that the NM has partial knowledge about the 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. 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, the static properties g and the domain knowledge ∇ if $\exists\sigma$ such that $p, g, \nabla \vdash \sigma \cdot condition$.

EXAMPLE. *In state p_0 the norm that forbids robots to move into occupied offices is instantiated three times as follows:*

$$\langle\mathcal{P}, move(R2, L1, d)\rangle \text{ where } \sigma = \{L2/d\}$$
$$\langle\mathcal{P}, move(R2, L1, a)\rangle \text{ where } \sigma = \{L2/a\}$$
$$\langle\mathcal{P}, move(R2, L1, a)\rangle \text{ where } \sigma = \{L2/e\}$$

Once the NM has determined which norm instances hold in a given situation, it has to check the actions of agents to determine which instances have been violated and which ones have been fulfilled.

Obligation Instance. In presence of partial knowledge about the actions performed by agents, the NM can only determine that an obligation instance has been fulfilled. If the NM knows all the actions performed by agents, then it can determine whether an obligation has been fulfilled or violated.

Definition 9. *Given an obligation instance $\langle\mathcal{O}, action'\rangle$ and a set of observed actions Act, then the obligation is defined as:*

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

Prohibition Instance. In presence of partial knowledge about the actions performed by agents, the NM can only determine that a prohibition instance has been violated. If the NM knows all the actions performed by agents then it can determine whether a prohibition has been fulfilled or violated.

Definition 10. *Given a prohibition instance $\langle\mathcal{P}, action'\rangle$ and a set of observed actions Act, then the prohibition is defined as:*

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

Finally, the set of discovered violations and fulfilments is used to identify those agents that have violated or fulfilled an instance.

EXAMPLE. *Taking into account the set of actions Act_0, the NM can* identify *that robot $r3$ has violated the instance $\langle\mathcal{P}, move(R2, L1, a)\rangle$, even though this forbidden action has not been observed by the NM. Specifically, there is $\sigma = \{R2/r3, L1/e\}$ such that $\sigma(move(R2, L1, a)) \in Act_0$. Besides that, the approximate reconstruction* discovers *that robot $r2$ has violated a prohibition instance though it doe snot know the exact action performed—recall that $D = \{move(r2, d, e)\}$. Had the NM not performed the proposed reconstruction processes, none of these violations would have been detected.*

IV. NM ALGORITHMS

Algorithm 1 contains the NM pseudocode. In each step, the NM observes the actions of agents and uses this information to update the current and the previous partial states (lines 4-9). If all the actions have not been observed in the previous state, then the NM executes the $reconstruction$ function to reconstruct unobserved actions (lines 11-14). Then, the $checkNorms$ function is executed to determine which norms have been violated and fulfilled in the previous state (line 15) according to Definitions 9 and 10.

Note that the NM code can be executed while actions are performed without delaying agents. Regarding the temporal cost of the algorithm executed by NMs, it is determined by the cost of the $reconstruction$ function, the implementations of which (full and approximate) are discussed below.

Algorithm 1 NM Algorithm

```
Require: $Ag, N, D, \nabla, g$
1:  $p_0 = \emptyset$                       ▷ $p_0$ is an empty conjunction of literals
2:  $t \leftarrow 0$
3:  while $true$ do
4:      $Act_t \leftarrow observeActions()$
5:      if $|Act_t| < |Ag|$ then
6:          $p_{t+1} \leftarrow post(Act_t)$
7:      else
8:          $p_{t+1} \leftarrow p_t^* \bigwedge eff(Act_t)$
9:      $p_t \leftarrow p_t \bigwedge pre(Act_t)$
10:     if $t > 0$ then
11:         if $|Act_{t-1}| < |Ag|$ then
12:             $TA \leftarrow Ag \setminus actors(Act_{t-1})$        ▷ Target Agents
13:             $D \leftarrow \emptyset$        ▷ Discovered violations and fuflilments
14:             $reconstruction(p_{t-1}, p_t, Act_{t-1}, TA, D)$
15:         $checkNorms(p_{t-1}, Act_{t-1})$
16:     $t \leftarrow t + 1$
```

Full Reconstruction (Algorithm 2). This pseudocode corresponds to the full reconstruction function. This function calls the function $search$ to search the actions of target agents (line 2). Then, for all the solutions found, the NM checks if they are consistent according to Definition 7 (lines 4-6). Finally, consistent solutions are used to extend the set of observed actions and the knowledge about the initial and final states (lines 7-14). The temporal cost of this algorithm is given by the cost of the $search$ function discussed below.

Algorithm 3 contains the pseudocode of the recursive $search$ function that computes all the sequences of consistent actions that may have been executed by the agents that have not been observed. It starts by checking that there is at least one target agent (line 2). If so, it identifies all actions that might have been executed by one target agent (lines 3-4). An action might have been executed if it is consistent according to the static properties, the domain knowledge, and the initial and final states. For each consistent action, it reconstructs the actions of the remaining agents recursively (lines 5-13). In the worst case, the temporal cost of this

Algorithm 2 Full Reconstruction Function

```
1:  function FULLRECONSTRUCTION($i, f, Act, TA, D$)
2:      $\mathcal{S}' \leftarrow search(i, f, Act, TA)$          ▷ Candidate Solutions
3:      $\mathcal{S} \leftarrow \emptyset$                              ▷ Consistent Solutions
4:      for all $S_j \in \mathcal{S}'$ do
5:          if $checkSolutionConsistency(Act, S_j, i, f)$ then
6:              $\mathcal{S} \leftarrow \mathcal{S} \cup S_j$
7:      $R \leftarrow \bigcap_{\forall S_i \in \mathcal{S}} S_i$
8:      if $R \neq \emptyset$ then
9:          $Act \leftarrow Act \cup R$
10:         $i \leftarrow i \bigwedge pre(R)$
11:         if $|Act| < |Ag|$ then
12:             $f \leftarrow f \bigwedge post(R) \bigwedge i^{\bullet}$
13:         else
14:             $f \leftarrow i^{*} \bigwedge f \bigwedge eff(Act)$
```

function is $O(|Ag|^{|D| \times I_D})$, where Ag is the set of agents, D is the set of action descriptions and I_D is the maximum number of instantiations per action. This situation arises when no action is observed and all actions are applicable for all agents.

Algorithm 3 Search Function

```
1:  function SEARCH($i, f, Act, TA$)
2:      if $TA \neq \emptyset$ then
3:          for all $d \in D$ do                   ▷ Identify consistent actions
4:              if $\exists \sigma$ : $checkActionConsitency(\sigma \cdot d, i, f) \wedge$
                $actor(\sigma \cdot d) \in TA$ then
5:                  $\alpha = actor(\sigma \cdot d)$
6:                  $i' \leftarrow i \bigwedge pre(\sigma \cdot d)$
7:                  $f' \leftarrow f \bigwedge post(\sigma \cdot d)$
8:                  $Act' \leftarrow Act \cup \sigma \cdot d$
9:                  $TA' \leftarrow TA \setminus \{\alpha\}$
10:                 $\mathcal{S} \leftarrow search(i', f', Act', TA')$
11:                 for all $S_i \in \mathcal{S}$ do
12:                     $S_i \leftarrow S_i \cup \sigma \cdot d$
13:                 return $\mathcal{S}$
14:     else
15:         return $\emptyset$
```

Approximate Reconstruction Function (Algorithm 4). This function calls the function $ApproximateSearch$ to search the applicable actions per each target agent (line 2). Then, the list of applicable actions per each agent is checked (lines 3-12). Specifically, if an agent may have executed one action only, then the NM knows that this action was executed and it updates the reconstructed action set (lines 3-5). Then, the set of observed actions and the knowledge about the initial and final states is updated (lines 6-12). Finally, discovered violations and fulfilments are calculated (lines 14-19). The temporal cost of this algorithm is given by the cost of the $ApproximateSearch$ function discussed below.

Algorithm 5 contains the pseudocode of the $ApproximateSearch$ function. It starts by initialising the list of applicable actions per agent (lines 3-4). Then it calculates the set of instances that are relevant to the initial state (line 7). The function calculates per each target agent the list of applicable actions that it may have executed (lines 9-11). Then, the list of applicable actions per each agent is checked (lines 12-18). Specifically, if an agent may have executed one action only, then the NM knows that this action was executed and it updates the list of applicable actions, the initial and final states, and retracts the agent from the target agents (lines 14-17). This process is repeated until there are no more target agents or the initial and final states remain unchanged. Then the set of instances that are relevant to the initial state is calculated (line 13). Finally, the list of applicable actions per agent is updated with actions of remaining target agents (lines 19-20). The temporal cost of this function is $O(|Ag|^2 \times |D| \times I_D)$.

Algorithm 4 Approximate Reconstruction Function

```
1:  function APPROXIMATERECONSTRUCTION($i, f, Act, TA, D$)
2:      $\widetilde{S} \leftarrow approximateSearch(i, f, Act, TA)$
3:      for all $\alpha \in TA$ do
4:          if $|\widetilde{S}_\alpha| = 1$ then
5:              $R \leftarrow R \cup S_\alpha$
6:      if $R \neq \emptyset$ then
7:          $Act \leftarrow Act \cup R$
8:          $i \leftarrow i \bigwedge pre(R)$
9:          if $|Act| < |Ag|$ then
10:             $f \leftarrow f \bigwedge post(R) \bigwedge i^{\circ}$
11:         else
12:             $f \leftarrow i^{*} \bigwedge f \bigwedge eff(Act)$
13:     $O, P \leftarrow calculateInstances(i)$
14:     for all $\alpha \in TA$ do
15:         if $|\widetilde{S}_\alpha| > 1$ then
16:             if $\nexists a \in \widetilde{S}_\alpha : \neg forbidden(P, a)$ then
17:                 $D \leftarrow D \cup a$          ▷ $a$ is an action from $\widetilde{S}_\alpha$
18:             else if $\nexists a \in \widetilde{S}_\alpha : \neg mandatory(O, a)$ then
19:                 $D \leftarrow D \cup a$          ▷ $a$ is an action from $\widetilde{S}_\alpha$
```

Algorithm 5 Approximate Search Function

```
1:  function APPROXIMATESEARCH($i, f, Act, TA$)
2:      $continue \leftarrow true$
3:      for all $\alpha \in TA$ do
4:          $\widetilde{S}_\alpha \leftarrow \emptyset$          ▷ List of approximate actions per agent
5:      while $continue \wedge TA \neq \emptyset$ do
6:          $continue \leftarrow false$
7:          for all $\alpha \in TA$ do
8:              $L_\alpha \leftarrow \emptyset$
9:          for all $d \in D$ do
10:             if $\exists \sigma$ : $checkActionConsitency(\sigma \cdot d, i, f) \wedge$
                $actor(\sigma \cdot d) \in TA$ then
11:                 $L_{actor(\sigma \cdot d)} \leftarrow L_{actor(\sigma \cdot d)} \cup \sigma \cdot d$
12:         for all $\alpha \in TA$ do
13:             if $|L_\alpha| = 1$ then
14:                 $\widetilde{S}_\alpha \leftarrow L_\alpha$
15:                 $i \leftarrow i \bigwedge pre(L_\alpha)$
16:                 $f \leftarrow f \bigwedge post(L_\alpha)$
17:                 $TA \leftarrow TA \setminus \{\alpha\}$
18:                 $continue \leftarrow true$
19:     for all $\alpha \in TA$ do
20:         $\widetilde{S}_\alpha \leftarrow L_\alpha$
21:     return $\widetilde{S}$
```

V. Evaluation

This section compares the performance of a NM with full reconstruction, a NM with approximate reconstruction and a traditional norm monitor —which is the method used in the majority of previous proposals [6], [23], [14], [24], [8]— that only considers the observed actions to detect violations; with respect to their capabilities to monitor norm compliance. We have evaluated our proposal in a case study, which allows us to contextualise the results and to give a meaningful interpretation to them; and in a series of random experiments, which allow us to evaluate our proposal under a wide range of different situations and parameter values.

A. *Case Study*

We implemented in Java a simulator of the paper example in which robots attend requests in offices connected through corridors. Compliance with the collision avoidance norm is controlled by a monitor that observes surveillance cameras. In each simulation, we generate corridors and cameras randomly. In each step of the simulation, each robot chooses randomly one applicable action to be executed. The simulation is executed 100 steps and repeated 100 times to support the findings. We conducted experiments in which the number of offices O took a random value within the $\llbracket 3, 500 \rrbracket$ interval and the number of robots R took a random value within the $\llbracket 2, 250 \rrbracket$ interval. Besides that, to be able to compare with the full NM, we also considered small scenarios only, in which the number of offices O takes a random value within $\llbracket 3, 10 \rrbracket$ and the number of robots R takes a random value within $\llbracket 2, 5 \rrbracket$, as the full reconstruction has an exponential cost and it is intractable for most of the cases with the default intervals.

1) Action Observability: To analyse the performance and scalability of monitors with respect to their capabilities to observe actions, we defined the number of corridors C as a random value within the $\llbracket O, O \times (O-1) \rrbracket$ interval and varied the ratio of cameras to corridors (action observability). Table I shows the percentage of violations detected per each type of monitor. The higher the ratio of cameras, the more actions are observed and the better the performance of all monitors. Moreover, the approximate NM offers on average a 39% performance improvement over a traditional monitor (i.e., it identifies 16% more violations plus a further 24% of discovered violations). That is, an approximate NM outperforms a traditional monitor with the same capabilities to observe actions. When compared to full NM in small scenarios ($O \in \llbracket 3, 10 \rrbracket$ and $R \in \llbracket 2, 5 \rrbracket$), approximate NM performs similarly. This is explained by the fact that there is a single norm in this scenario, actions have no concurrency conditions, and the preconditions and postconditions of actions are disjoint. In this circumstances, the approximate reconstruction process reconstructs actions similarly to the full reconstruction[14].

Cameras Ratio	Traditional Monitor	Approximate NM Identify+Discover
0%	0%	0+0%
20%	11%	14+9%
40%	31%	40+12%
60%	55%	71+10%
80%	78%	91+4%
100%	100%	100+0%

$O \in \llbracket 3, 500 \rrbracket$ and $R \in \llbracket 2, 250 \rrbracket$

Cameras Ratio	Traditional Monitor	Full NM	Approximate NM Identify+Discover
0%	0%	0%	0+0%
20%	16%	32%	32+6%
40%	32%	68%	67+5%
60%	56%	88%	88+3%
80%	76%	99%	99+0%
100%	100%	100%	100+0%

$O \in \llbracket 3, 10 \rrbracket$ and $R \in \llbracket 2, 5 \rrbracket$

TABLE I: Action Observability Experiment

[14]Note that full reconstruction does not guarantee completeness.

2) Action Instantiations: To analyse the performance and scalability of monitors with respect to agent capabilities to execute actions (i.e., the number of instantiations per action), we varied the ratio of corridors[15] (e.g., a ratio of 0% means $C = O$) and defined the number of cameras as a random value within the $\llbracket 0, C \rrbracket$ interval. Table II shows the results of this experiment. The approximate NM offers on average a 43% performance improvement over a traditional monitor (i.e., it identifies 29% more violations plus a further 14% of discovered violations). That is, given the same number of possible instantiations per action, an approximate NM outperforms a traditional monitor. Besides, we can see that, as in the previous experiment, the approximate NM performs similarly to the full NM. In particular, when the ratio of corridors is higher than 0%, agents are capable of executing different actions and the reconstruction process becomes more complex, which decreases the performance of full and approximate NMs. However, full and approximate NMs noticeably outperform the traditional monitor regardless of the ratio of corridors.

Corridors Ratio	Traditional Monitor	Approximate NM Identify+Discover
0%	51%	98+0%
20%	48%	55+6%
40%	48%	56+7%
60%	41%	47+6%
80%	49%	56+13%
100%	42%	49+7%

$O \in \llbracket 3, 500 \rrbracket$ and $R \in \llbracket 2, 250 \rrbracket$

Corridors Ratio	Traditional Monitor	Full NM	Approximate NM Identify+Discover
0%	52%	99%	99+0%
20%	59%	80%	79+3%
40%	55%	74%	74+3%
60%	51%	70%	69+4%
80%	55%	68%	68+5%
100%	57%	70%	69+4%

$O \in \llbracket 3, 10 \rrbracket$ and $R \in \llbracket 2., 5 \rrbracket$

TABLE II: Action Instantiations Experiment

B. *Random Experiments*

We implemented a simulator in Java in which there is a set of agents that perform actions in a monitored environment as defined below. In particular, our simulator does not model a specific scenario; rather it creates a different scenario in each simulation (i.e., generating randomly agent capabilities, the environment properties, actions and norms). As in the previous experiments, we have considered big and small scenarios. In particular, the number of agents G in small scenarios took a random value within the $\llbracket 1, 5 \rrbracket$ interval, whereas in big scenarios G took a random value within the $\llbracket 1, 500 \rrbracket$ interval. The number of actions A took a random value within the $\llbracket 1, 50 \rrbracket$ interval. Again, the simulation is executed 100 steps and repeated 1000 times to ensure that the values of the simulation parameters range over possible values[16].

Agent Definition. We modelled different types of agents with different capabilities to perform actions. In particular, the set of actions available to each agent depends on the function/s assumed by each agent in a particular simulation. To model these capabilities, a set of roles is created at the beginning of each simulation. Specifically, the number of roles created took a random value within the $\llbracket 1, A \rrbracket$ interval. For each role a subset of the actions are randomly selected as the role capabilities; i.e., all agents enacting this role are able to

[15]Recall that C takes values within the $\llbracket O, O \times (O-1) \rrbracket$ interval.

[16]Note that in the random experiments there are more simulation parameters than in the case-study simulator and a higher number of repetitions is required to support the findings.

perform these actions[17]. To avoid that all roles have similar capabilities, which would lead to simulations populated by homogeneous agents, 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 a 10% of the actions). At the beginning of each simulation, each agent is defined as enacting a random subset of the roles. In each step of the simulation, each agent selects randomly one action among the available actions that it can execute in the current state.

Environment Definition. In the simulator, the environment is described in terms of different situations or states of affairs that can be true or false. Each one of these states of affairs is represented by a grounded proposition. Thus, the state of the environment is defined in terms of a set of propositions. For simplicity, we assumed that these propositions are independent (i.e, propositions are not logically related). In our simulations, 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[18]). Besides that, there is a set of grounded atomic formulas describing the roles played by agents and the actions that can be performed by each role. The relationship between agents and roles is formally represented by a binary predicate ($play$). Specifically, the expression $play(g, r)$ describes the fact that the agent identified by g enacts the role identified by r. Similarly, relationship between roles and actions is formally represented by a binary predicate ($capable$). Specifically, the expression $capable(a, r)$ describes the fact that agents enacting role r are capable of performing the action identified by a. For simplicity, we assume that the roles enacted by the agents and the role capabilities are static properties of the environment.

Action Definition. Actions allow agents to change the state of the environment. At the beginning of each simulation, a set of actions is randomly generated. For each action $\langle name, pre, con, post \rangle$ the elements are defined as follows: $name$ is initialised with a sequential identifier a; pre is defined as $\{play(A, R), capable(R, a), p_1, ..., p_n\}$ where the elements $p_1, ..., p_n$ are randomly selected from the proposition set; con is defined as $\{a_1(A_1, R_1), ..., a_m(A_m, R_m)\}$, where each a_i is an action randomly selected from the action set such that $a_i \neq a$ and A_i, R_i are free variables representing the agent performing the action and the role capable of performing this action, respectively; and $post$ is defined as $\{p_1, ..., p_k\}$ where each p_i is a proposition randomly selected from the proposition set. To avoid that actions have too many constraints, which would be unrealistic and make actions to be only executed on few situations, the number of propositions in pre and $post$ takes a random value within the $[\![1, \lceil 0.1 * P \rceil]\!]$ interval. Similalry, the number of actions in con takes a random value within the $[\![0, \lceil 0.1 * A \rceil]\!]$ interval.

Besides these actions, a NOP action, which has no effect on the environment, was created. To maximise the number of actions executed in the simulations, which may entail more violations and fulfilments, we defined that the NOP action can only be executed by agents when none of their available actions can be executed. However, similar results would have been obtained if this condition was relaxed[19]. Our simulator models scenarios where the NOP action can always be observed. This is the case in many real domains such as Intrusion Detection Systems or Autonomous Systems, where it it not always possible to analyse the data (e.g., the packages) sent by agents (e.g., hosts) to infer the actions performed, but it is always possible to know which agents have performed an action (i.e., which agents have sent packages).

Norm Definition. Agents' actions are regulated by a set of norms. At the beginning of each simulation, a set of norms is randomly created. 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). For each action $\langle deontic, condition, action \rangle$ the elements are defined as follows: $deontic$ is randomly initialised with a deontic operator; $condition$ is defined as $\{p_1, ..., p_k\}$ where each p_i is a proposition randomly selected from the proposition set; and $action$ is randomly initialised with an action. To allow norms to be instantiated, the number of propositions in $condition$ takes a random value within the $[\![0, \lceil P * 0.1 \rceil]\!]$ interval.

1) Action Observability: To analyse the performance and scalability of monitors with respect to their capabilities to observe actions, we varied the observation probability. Tables III and IV show the percentage of detected fulfilments and violations, respectively. Again, the approximate NM offers a significant performance improvement over a traditional monitor; i.e., the approximate NM offers on average a 74% performance improvement over a traditional monitor. When compared to full NM in small scenarios ($A \in [\![1, 50]\!]$ and $G \in [\![1, 5]\!]$), the full NM offers on average a 21% performance improvement over an approximate NM. This is explained by the fact that this experiment is more complex than the case study; i.e., there are several norms (both prohibition and obligation norms), actions have concurrent conditions and actions may have conflicting preconditions and postconditions (i.e., conditions that are defined over the same propositions). Note that the traditional monitor detects violations and fulfilments even when the observation probability is 0%. These detections correspond to situations in which none of the agents can execute any action (i.e., all agents execute the NOP action) which leads to the fulfilment of prohibition instances and the violation of obligation instances. This phenomenon is more frequent in case of small scenarios since the lower the number of agents, the higher the probability that all agents cannot execute any action.

2) Action Possibilities: To analyse the performance and scalability of monitors with respect to agent capabilities to execute actions (i.e., the number of available actions), we defined the observation probability as a random value within the $[0, 100\%]$ interval and we varied the number of actions. Tables V and VI show the percentage of detected fulfilments and violations, respectively. In this experiment, the more actions, the more complex the reconstruction problem is. As a consequence, the improvement offered by an approximate

[17]This condition has been formulated in action preconditions as explained below.

[18]Note that an action can change the truth value of several propositions.

[19]Note that the capabilities of monitors to detect violations and fulfilments do not depend on the fact that agents are allowed to perform the NOP action in any situation.

Observ. Prob.	Traditional Monitor	Approximate NM Identify+Discover
0%	2%	35+7%
20%	18%	47+6%
40%	35%	62+4%
60%	50%	72+3%
80%	66%	80+2%
100%	100%	100+0%

$A \in \llbracket 1, 50 \rrbracket$ and $G \in \llbracket 1, 500 \rrbracket$

Observ. Prob.	Traditional Monitor	Full NM	Approximate NM Identify+Discover
0%	20%	46%	34+1%
20%	23%	51%	39+1%
40%	31%	57%	45+1%
60%	43%	69%	56+0%
80%	58%	79%	70+0%
100%	100%	100%	100+0%

$A \in \llbracket 1, 50 \rrbracket$ and $G \in \llbracket 1, 5 \rrbracket$

TABLE III: Fulfilments Detected in the Action Observability Experiment

Observ Prob.	Traditional Monitor	Approximate NM Identify+Discover
0%	2%	36+8%
20%	18%	50+6%
40%	35%	62+6%
60%	49%	70+2%
80%	65%	78+2%
100%	100%	100+0%

$A \in \llbracket 1, 50 \rrbracket$ and $G \in \llbracket 1, 500 \rrbracket$

Observ Prob.	Traditional Monitor	Full NM	Approximate NM Identify+Discover
0%	20%	44%	33+1%
20%	23%	50%	37+0%
40%	32%	56%	45+0%
60%	42%	68%	55+0%
80%	56%	78%	69+0%
100%	100%	100%	100+0%

$A \in \llbracket 1, 50 \rrbracket$ and $G \in \llbracket 1, 5 \rrbracket$

TABLE IV: Violations Detected in the Action Observability Experiment

NM over a traditional monitor decreases as the number of actions increases. However, the approximate NM still offer on average a 56% performance improvement over a traditional monitor. When the number of actions is very high (e.g., when the number of actions is 128 in small scenarios), then action preconditions become very complex and most of the times the NOP action is executed by all agents, which entails that the all monitors obtain a good performance. We can see that, as in the previous experiment, the approximate NM performs slightly worse than the full NM (i.e., the full NM offers on average a 15% performance improvement over an approximate NM). However, full and approximate NMs noticeably outperform the traditional monitor regardless of the number of actions.

Actions	Traditional Monitor	Approximate NM Identify+Discover
2	52%	95+8%
8	49%	75+1%
32	32%	48+0%
128	29%	40+0%

$G \in \llbracket 1, 500 \rrbracket$

Actions	Traditional Monitor	Full NM	Approximate NM Identify+Discover
2	69%	99%	95+3%
8	47%	75%	71+0%
32	38%	63%	48+0%
128	53%	83%	57+0%

$G \in \llbracket 1, 5 \rrbracket$

TABLE V: Fulfilments Detected in the Action Possibilities Experiment

Actions	Traditional Monitor	Approximate NM Identify+Discover
2	53%	93+5%
8	49%	76+6%
32	35%	50+1%
128	31%	39+0%

$G \in \llbracket 1, 500 \rrbracket$

Actions	Traditional Monitor	Full NM	Approximate NM Identify+Discover
2	68%	99%	95+2%
8	48%	76%	72+1%
32	38%	62%	47+0%
128	56%	85%	59+0%

$G \in \llbracket 1, 5 \rrbracket$

TABLE VI: Violations Detected in the Action Possibilities Experiment

C. Summary

The conclusions of our evaluation are threefold:

1) Both approximate and full reconstruction processes are more effective (i.e., detect more norm violations and fulfilments) than traditional monitoring approaches regardless of the scenario complexity (i.e., action possibilities and observability). Both in the case study and in the random experiments our algorithms improved significantly the percentage of violations and fulfilments detected.
2) Approximate reconstruction is slighting less effective than full reconstruction. In the case study, where a single prohibition norm was monitored; the approximate NM obtained almost the same results as the full NM. In our random experiments, where several prohibition and obligation norms were monitored, the full NM offered an average improvement of a 18% over an approximate NM.
3) Approximate reconstruction is scalable with the scenario size (i.e., the number of agents and actions to be monitored). In particular, our experiments demonstrate that the approximate algorithm can be used to monitor a large number of agents (we simulated scenarios with up to 500 agents), actions (we simulated scenarios with up to 128 actions), and norms (we simulated scenarios with up to 128 norms).

VI. Related Work

Previous work on norms for regulating MAS proposed control mechanisms for norms to have an effective influence on agent behaviours [15]. These *control* mechanisms are classified into two main categories [15]: *regimentation* mechanisms, which make the violation of norms impossible; and *enforcement* mechanisms, which are applied after the detection of norm violations and fulfilments, reacting upon them.

Regimentation mechanisms prevent agents from performing forbidden actions (vs. force agents to perform obligatory actions) by mediating access to resources and the communication channel, such as Electronic Institutions (EIs) [12]. However, the regimentation of all actions is often difficult or impossible. Furthermore, it is sometimes preferable to allow agents to make flexible decisions about norm compliance [7]. In response to this need, *enforcement* mechanisms were developed. Proposals on the enforcement of norms can be classified according to the entity that monitors whether norms are fulfilled or not. Specifically, norm compliance can be monitored by either agents themselves or the underlying infrastructure may provide monitoring entities.

Regarding *agent monitoring*, this approach is characterized by the fact that norm violations and fulfilments are monitored by agents that are involved in an interaction [30], [9], or other agents that observe an interaction in which they are not directly involved [28], [10], [32]. The main drawback of proposals based on agent monitoring is the fact that norm monitoring and enforcement must be implemented by agent programmers.

Regarding *infrastructural monitoring*, several authors proposed developing entities at the infrastructure level that are in charge of both monitoring and enforcing norms. Cardoso & Oliveira [6] proposed an architecture in which the monitoring and enforcement of norms is made by a single institutional entity. This centralized implementation represents a performance limitation when dealing with a considerable number of agents. To address the performance limitation of centralized approaches, distributed mechanisms for an institutional enforcement of norms were proposed in [23], [14], [24], [8].

All of the aforementioned proposals on monitoring assume that monitors have complete observational capabilities. Exception to these approaches is the recent work of Bulling et al. [5] and Alechina et al. [2]. In [5], the partial observability problem is addressed combining different norm monitors to build ideal monitors (i.e., monitors that together are able to detect the violation of a given set of norms). In [2], the authors propose to synthesise an approximate set of norms that can be monitored given the observational 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 reconstruct unobserved actions.

Our approach is also related to planning, where methods (e.g., POMDPs [18]) for choosing optimal actions in partially observable environments have been proposed. A major difference between these proposals and our proposal is that NMs do not perform practical reasoning, i.e., they do not try to optimise or achieve a practical goal. Instead, NMs perform both deductive and abductive reasoning [25] to reason from observed actions to reach a conclusion about the state of the world, and to infer unobserved actions.

VII. Conclusion

In this paper, we propose information models and algorithms for monitoring norms under partial action observability by reconstructing unobserved actions from observed actions using two different reconstruction processes: full and approximate. Our experiments demonstrate that both reconstruction processes detect more norm violations than traditional monitoring approaches. Approximate reconstruction performs slightly worse than full reconstruction, whereas its computational cost is much cheaper, making it suitable to be applied in practice.

The reconstruction algorithms proposed in this paper can be applied to several domains that require action monitoring; from normative MAS [31], to intrusion detection systems [16], to control systems [26] and to intelligent surveillance systems [17]. As future work, we plan to investigate domain-dependent approximations that could speed up action reconstruction even further.

References

[1] N. Alechina, M. Dastani, and B. Logan. Reasoning about normative update. In *Proc. of IJCAI*, 2013.

[2] N. Alechina, M. Dastani, and B. Logan. Norm approximation for imperfect monitors. In *Proc. of AAMAS*, pages 117–124, 2014.

[3] T. Bass. Intrusion detection systems and multisensor data fusion. *Communications of the ACM*, 43(4):99–105, 2000.

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

[5] N. Bulling, M. Dastani, and M. Knobbout. Monitoring norm violations in multi-agent systems. In *Proc. of AAMAS*, pages 491–498, 2013.

[6] H. Cardoso and E. Oliveira. Institutional reality and norms: Specifying and monitoring agent organizations. *International Journal of Cooperative Information Systems*, 16(1):67–95, 2007.

[7] C. Castelfranchi. Formalising the informal? Dynamic social order, bottom-up social control, and spontaneous normative relations. *Journal of Applied Logic*, 1(1-2):47–92, 2003.

[8] N. Criado, E. Argente, P. Noriega, and V. Botti. Manea: A distributed architecture for enforcing norms in open mas. *Engineering Applications of Artificial Intelligence*, 26(1):76–95, 2013.

[9] A. Daskalopulu, T. Dimitrakos, and T. Maibaum. Evidence-based electronic contract performance monitoring. *Group Decision and Negotiation*, 11(6):469–485, 2002.

[10] A. P. de Pinninck, C. Sierra, and W. M. Schorlemmer. A multiagent network for peer norm enforcement. *Journal of Autonomous Agents and Multi-Agent Systems*, 21(3):397–424, 2010.

[11] F. Dignum. Autonomous agents with norms. *Artificial Intelligence and Law*, 7(1):69–79, 1999.

[12] M. Esteva, B. Rosell, J. A. Rodríguez-Aguilar, and J. L. Arcos. Ameli: An agent-based middleware for electronic institutions. In *Proc. of AAMAS*, pages 236–243, 2004.

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

[14] D. Gaertner, A. Garcia-Camino, P. Noriega, J.-A. Rodriguez-Aguilar, and W. Vasconcelos. Distributed norm management in regulated multiagent systems. In *Proc. of AAMAS*, pages 624–631, 2007.

[15] D. Grossi, H. Aldewereld, and F. Dignum. Ubi lex, ibi poena: Designing norm enforcement in e-institutions. In *COIN II*, pages 101–114. Springer, 2007.

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

[17] C.-M. Huang and L.-C. Fu. Multitarget visual tracking based effective surveillance with cooperation of multiple active cameras. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41(1):234–247, 2011.

[18] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Journal of Artificial Intelligence*, 101(1):99–134, 1998.

[19] F. López y López, M. Luck, and M. dInverno. A normative framework for agent-based systems. *Computational & Mathematical Organization Theory*, 12(2-3):227–250, 2006.

[20] E. Lorini. On the logical foundations of moral agency. In *Deontic Logic in Computer Science*, pages 108–122. Springer, 2012.

[21] F. Meneguzzi, S. Modgil, N. Oren, S. Miles, M. Luck, and N. Faci. Applying electronic contracting to the aerospace aftercare domain. *Engineering Applications of Artificial Intelligence*, 25(7):1471–1487, 2012.

[22] R. Micalizio, P. Torasso, and G. Torta. On-line monitoring and diagnosis of multi-agent systems: a model based approach. In *Proc. of ECAI*, volume 16, page 848, 2004.

[23] N. 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.

[24] 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.

[25] G. Paul. Approaches to abductive reasoning: an overview. *Journal of Artificial Intelligence Review*, 7(2):109–152, 1993.

[26] C. Rieger, D. Scheidt, and W. Smart. Guest editorial: Introduction to the special issue on resilient control architectures and systems. *IEEE Transactions on Cybernetics*, 44(11):1994–1996, Nov 2014.

[27] S. Sadiq, G. Governatori, and K. Namiri. Modeling control objectives for business process compliance. In *Business process management*, pages 149–164. Springer, 2007.

[28] S. Sen and S. Airiau. Emergence of norms through social learning. In *Proc. of IJCAI*, pages 1507–1512, 2007.

[29] W. Vasconcelos, M. J. Kollingbaum, and T. J. Norman. Resolving conflict and inconsistency in norm-regulated virtual organizations. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 91. ACM, 2007.

[30] M. Venkatraman and M. Singh. Verifying compliance with commitment protocols. *Journal of Autonomous Agents and Multi-Agent Systems*, 2(3):217–236, 1999.

[31] C. Yu, M. Zhang, and F. Ren. Collective learning for the emergence of social norms in networked multiagent systems. *IEEE Transactions on Cybernetics*, 44(12):2342–2355, Dec 2014.

[32] W. Zeng and M.-Y. Chow. Resilient distributed control in the presence of misbehaving agents in networked control systems. *IEEE Transactions on Cybernetics*, 44(11):2038–2049, Nov 2014.