Norm Approximation for Imperfect Monitors

Natasha Alechina
School of Computer Science
University of Nottingham
Nottingham NG8 1BB, UK
nza@cs.nott.ac.uk

Mehdi Dastani
Department of Information and
Computing Sciences
Universiteit Utrecht
The Netherlands
M.M.Dastani@uu.nl

Brian Logan
School of Computer Science
University of Nottingham
Nottingham NG8 1BB, UK
bsl@cs.nott.ac.uk

ABSTRACT

In this paper, we consider the runtime monitoring of norms with imperfect monitors. A monitor is imperfect for a norm if it has insufficient observational capabilities to determine if a given execution trace of a multi-agent system complies with or violates the norm. One approach to the problem of imperfect monitors is to enhance the observational capabilities of the normative organisation. However this may be costly or in some cases impossible. Instead we show how to synthesise an approximation of an 'ideal' norm that can be perfectly monitored given a monitor, and which is optimal in the sense that any other approximation would fail to detect at least as many violations of the ideal norm. We give a logical analysis of (im)perfect monitors. We state the computational complexity of the norm approximation problem, and give an optimal algorithm for generating optimal approximations of norms given a monitor.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multi-agent systems*

Keywords

Norms; Monitoring

1. INTRODUCTION

Norms have been widely proposed as a means of coordinating and regulating the behaviours of agents within a multi-agent system (MAS). In a normative multiagent system, the interaction between agents and their environment is governed by a *normative organisation* specified by a set of norms (obligations and prohibitions). Individual agents update the state of the environment by performing actions. The role of the normative organisation is to continuously evaluate the state updates resulting from agent actions with respect to the norms to 1) determine any new obligations to be fulfilled or prohibitions that should not be violated (termed detached obligations and prohibitions), 2) check if any previously detached norms are obeyed or violated in the current state, and 3) impose sanctions when norms are violated. This continuous process is often implemented by a so-called control cycle [9].

For a system of norms to have their intended effect, it must be possible for the normative organisation to monitor the environment updates to determine when norms are detached, and when a detached norm is obeyed or violated. In general, previous work on normative organisations has implicitly assumed that the environment is fully observable, and hence can be perfectly monitored. For example, in determining whether a set of norms will have their intended effect, [3] assumes the environment can be perfectly monitored. In practice, this assumption may not hold. Norms specify the ideal behaviours of an agent or agents independent of the realisation of these behaviours in a particular MAS. However the environment of a particular multi-agent system may not be fully observable, or some facts may be observable only at great cost. For example, consider a university norm 'feedback should be returned to a student by the deadline, otherwise a reminder is sent to the lecturer'. It is difficult to automatically monitor whether proper feedback has been sent, but it is possible to check that a message contains a grade, a minimal amount of text for each assessed component etc. As another example, consider a long motorway where the norms to be enforced are driving regulations, such as speed limit etc. It is too expensive to position monitoring equipment so that the norms are monitored along the entire motorway. In such environments, it is impossible or infeasible for the normative organisation to monitor (and so enforce) the ideal norms specified by the designer of the normative system.

One way of solving this problem is to change the 'ideal' norms such that all violations of the new norm become detectable. This raises further problems: which of the possible modifications or approximations of the original 'ideal' norm is 'closest' to the intent of the designer of the normative system. Continuing with the motorway example, one possible choice is between using sparse but accurate speed cameras (and essentially approximating the speed limit norm to mean that speeding is not permitted in the neighbourhood of the speed camera) or cheaper, more numerous but less accurate cameras (which means that the norm is approximated to prohibit speeding in more places, but adjusting the precise value of the speed limit).

Such approximate norms are important for several reasons. Firstly, norm-aware agents, e.g., [2], need to know which norms are actually in force in a normative MAS in order to make a rational decision whether to comply with a norm. Secondly, the designer of a normative MAS needs to know which approximations of norms correspond to the behaviours that can actually be monitored by the monitoring capabilities of a particular normative organisation in order to determine whether the realisable approximation of a set of ideal norms will have an effect that is sufficiently close to the ideal system behaviour specified by the ideal norms.

The key contribution of this paper is to define a principled approach to norm approximation. Given a set of 'ideal' norms, a set of observable properties and some relationships between observ-

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

able properties and norms, we show how to automatically synthesise optimal (in the sense of minimising undetected violations) approximations of the ideal norms defined in terms of the observable properties. We show that the procedure is also optimal in the sense that its complexity matches the complexity of the norm approximation problem.

The structure of the paper is as follows. In section 2 we define conditional norms and (perfect) monitors. In section 3 we introduce an example. In section 4 we introduce imperfect monitors which do not have perfect observational capabilities. In section 5 we provide an optimal procedure for norm approximation. We survey related work and conclude in section 6.

2. CONDITIONAL NORMS

We focus on *conditional norms* with deadlines and sanctions [16], as these represent a reasonable compromise between expressive power and ease of reasoning about norms and monitoring.

DEFINITION 1 (NORMS). *Let cond, ϕ, d and san be boolean combinations of propositional variables from some propositional language L_N. A* conditional obligation *is represented by a tuple* $(cond, O(\phi), d, san)$ *and a* conditional prohibition *is represented by a tuple* $(cond, P(\phi), d, san)$. *A norm set N is a set of conditional obligations and conditional prohibitions.*

We assume that the possible behaviours of the agents are represented by a transition system. A transition system M is a tuple (S, R, V, s_I) where S is a set of states, $s_I \in S$ is the initial state, R is the transition relation (intuitively, corresponding to possible actions the agents can perform) and V is a labelling of states with propositional variables. Conditional norms are evaluated on runs of the transition system. A conditional norm $n = (cond, Z(\phi), d, san)$, where Z is O or P, is *detached* in a state satisfying its condition $cond$. Detached norms persist as long as they are not obeyed or violated, even if the triggering condition of the corresponding conditional norm does not hold any longer. A detached obligation $(cond, O(\phi), d, san)$ is *obeyed* if no state satisfying d is encountered before execution reaches a state satisfying ϕ, and *violated* if a state satisfying d is encountered before execution reaches a state satisfying ϕ. Conversely, a detached prohibition $(cond, P(\phi), d, san)$ is obeyed if no state satisfying ϕ is encountered before execution reaches a state satisfying d, and violated if a state satisfying ϕ is encountered before execution reaches a state satisfying d. If a detached norm is violated in a state s, the sanction corresponding to the norm is applied (becomes true) in s. We say a detached norm is annulled in a state s' immediately after a state s in which the norm is obeyed or violated, unless the same norm is detached again in s'. As our focus in this paper is on the monitoring of norms, in the interests of readability we abstract from sanctions and omit the san component from the representation of norms below.

In what follows, we focus on monitoring for violations of conditional norms: that is, determining if a particular run of the system violates a conditional norm. Algorithm 1 illustrates how a conditional obligation can be monitored on a finite path ρ. The function VIOL returns *true* if the conditional norm $(cond, O(\phi), d)$ is violated in a state $\rho[i]$ at position i on ρ and *false* otherwise. The algorithm for monitoring for violations of conditional prohibitions is similar and is illustrated in Algorithm 2.

A (perfect) monitor for a conditional norm is a device which takes a finite path and a conditional norm and returns true if the path violates the norm and false if it does not violate the norm, in other words it implements something like the algorithms above. Note that in order to be able to implement the algorithms, the monitor needs to be able to perform the boolean queries $cond$, ϕ and d on the states. This means that the states must be perfectly observable as far as these queries are concerned.

Algorithm 1 Violation of a Conditional Obligation

```
bool det ← false
function VIOL($(cond, O(\phi), d), \rho$)
    for $i \leftarrow 1$ to $\rho.length$ do
        if $\rho[i] \models cond$ then
            $det \leftarrow true$
        end if
        if $det$ then
            if $\rho[i] \models d \wedge \rho[i] \not\models \phi$ then
                return $true$
            else if $s \models \phi$ then
                $det \leftarrow false$
            end if
        end if
    end for
    return $false$
end function
```

Algorithm 2 Violation of a Conditional Prohibition

```
bool det ← false
function VIOL($(cond, O(\phi), d), \rho$)
    for $i \leftarrow 1$ to $\rho.length$ do
        if $\rho[i] \models cond$ then
            $det \leftarrow true$
        end if
        if $det$ then
            if $\rho[i] \models \phi \wedge \rho[i] \not\models d$ then
                return $true$
            else if $s \models d$ then
                $det \leftarrow false$
            end if
        end if
    end for
    return $false$
end function
```

Another way to give precise meaning to violations of conditional norms is to express them as $PLTL$ formulas. First we need to introduce some background on $PLTL$. We follow the semantics of $PLTL$ on finite traces adopted in [12]. Given a transition system $M = (S, R, V, s_I)$ with an initial state $s_I \in S$, a *path* through M is a sequence $s_0, s_1, s_3, \ldots$ of states such that $s_i R s_{i+1}$ for $i = 0, 1, \ldots$ and the first state is $s_0 = s_I$. In this paper, we consider finite paths. We denote paths by $\rho, \rho', \ldots$, and the state at position i on ρ by $\rho[i]$. The syntax of $PLTL$ formulas is defined relative to a set of propositional variables Π as follows:

$$p \in \Pi \mid \neg\phi \mid \phi \wedge \psi \mid \mathcal{Y}\phi \mid \phi \mathcal{S} \psi$$

where $\mathcal{Y}$ means previous state unless we are in the first state of the sequence, in which case it means the current state, and $\mathcal{S}$ stands for since. The truth definition for formulas is given relative to M, a path ρ and the state at position i on ρ:

$M, \rho, i \models p$ iff $p \in V(\rho[i])$

$M, \rho, i \models \neg\phi$ iff $M, \rho, i \not\models \phi$

$M, \rho, i \models \phi \wedge \psi$ iff $M, \rho, i \models \phi$ and $M, \rho, i \models \psi$

$M, \rho, i \models \mathcal{Y}\phi$ iff either $i > 0$ and $M, \rho, i-1 \models \phi$, or $i = 0$ and $M, \rho, i \models \phi$.

$M, \rho, i \models \phi \mathcal{S} \psi$ iff $\exists j \leq i$ such that $M, \rho, j \models \psi$ and $\forall k : j < k \leq i$, $M, \rho, k \models \phi$

Now we are ready to give a translation in $PLTL$ of the violation conditions of norms (essentially the same translation as given in [3]):

DEFINITION 2 (VIOLATION FORMULA). *A* violation formula *for a conditional obligation* $(c, O(\phi), d)$ *is*

$$d \wedge \neg\phi \wedge ((\mathcal{Y}\,(\neg\phi \wedge \neg d)\,\mathcal{S}\,(c \wedge \neg\phi \wedge \neg d)) \vee c)$$

A violation formula *for a conditional prohibition* $(c, P(\phi), d)$ *is*

$$\neg d \wedge \phi \wedge ((\mathcal{Y}\,(\neg\phi \wedge \neg d)\,\mathcal{S}\,(c \wedge \neg\phi \wedge \neg d)) \vee c)$$

We will denote a violation formula for a conditional norm n by $v(n)$. A norm n is violated on a path ρ if, and only if, $v(n)$ holds in some state on ρ. An obligation $(c, O(\phi), d)$ is violated on ρ if in some state the deadline description d is true, and it holds that some time in the past the condition of the norm became true and since then the obligation description ϕ has not been true (and is not true in the current state). A prohibition $(c, P(\phi), d)$ is violated on ρ if somewhere on ρ there exists a state where the prohibition description ϕ is true, and some time in the past the condition of the norm became true and since then the deadline description has not become true.

THEOREM 1. *A conditional norm* n *is violated on a path* ρ *in model* M *if and only if there exists a state* $\rho[i]$ *on* ρ *such that* $M, \rho, i \models v(n)$.

PROOF. Obvious from the conditions of norm violation given by Algorithms 1 and 2. □

Clearly, conditional norms are not as expressive as arbitrary $PLTL$ formulas.

THEOREM 2. *The language of violation formulas is strictly less expressive than the full* $PLTL$ *language.*

PROOF. In order to prove this, we exhibit an operation on runs which preserves violation formulas but does not preserve arbitrary $PLTL$ formulas. Consider a run ρ and an operation dup which duplicates states on ρ. Clearly, if a formula of the form $v(n)$ is true on ρ, then it is true on $dup(\rho)$. However, arbitrary $PLTL$ formulas are not preserved under this operation. □

Given that conditional norms are less expressive than $PLTL$ formulas, one might ask why not use $PLTL$ for expressing norms? In addition to the simplicity of Algorithms 1 and 2 and intuitiveness of the meaning of the norms, there is another consideration which is important for the topic of this paper: conditional norms are easier to approximate under restricted observability of states than arbitrary $PLTL$ formulas.[1]

3. EXAMPLE

In this section, we introduce a running example to illustrate the notions introduced in the paper. Consider the transition system shown in Figure 1 which represents possible behaviours of a car, where the propositional variables have the following meanings:

- p_i: the car is identified at position i.
- s_x: the speed of the car is x mph.

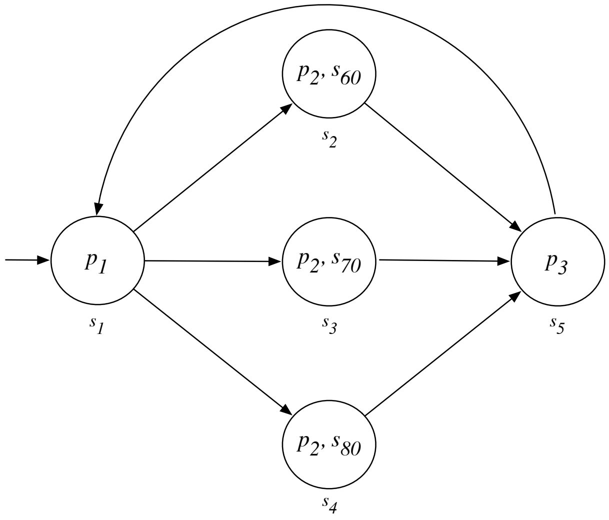


Figure 1: Example transition system M_t.

Consider the norm prescribing that a car should have a maximum speed of 60 mph between positions 1 and 3 on a ringway road. Assuming that the only possible values for the speed are 60, 70 and 80 mph, this conditional norm can be represented as a conditional prohibition $n_1 = (p_1, P(p_2 \wedge (s_{70} \vee s_{80})), p_3)$ or as a conditional obligation $n_2 = (p_1, O(p_2 \wedge s_{60}), p_3)$. Clearly, both norms are violated on the paths s_1, s_3, s_5 and s_1, s_4, s_5. On s_1, s_3, s_5, prohibition n_1 is detached in s_1 and violated in s_3. On the same path, obligation n_2 is detached in s_1 and violated in s_5.

4. IMPERFECT MONITORS

In previous sections, we considered monitoring norms under the condition of perfect observability. We could always evaluate boolean queries used in the norms in each state. However, in many situations, such perfect observability may not be possible. For example, we may only have a speed camera that can distinguish between speeds of 70 mph and less and speeds over 70 mph. To study monitoring under partial observability, we generalise the notion of a monitor to include a set of queries which the monitor can ask of a state. 'Standard' monitors then become a special case of our monitors, where the set of possible queries includes all possible queries definable in the language of conditional norms.

We start with a language L which is given as a set of formulas; we don't assume anything about the formulas apart from that they can be evaluated in the states of the system to true or false. Intuitively, the formulas in L define the set of possible observations or queries with a yes or no answer. For example, a formula may say that an object in front is a car, or that it is moving at a speed in excess of 100 mph, or that it is a red Lamborghini, or that there is no person in the driving seat, etc.

DEFINITION 3 (MONITOR). *A monitor* m_Φ *(where* $\Phi \subseteq L$ *is a finite set of formulas) is a function which given a finite path* ρ *and a norm* n*, returns true, false or undefined.*

Intuitively, m_Φ implements a monitoring algorithm for conditional norms, and it uses only the boolean combinations of queries from Φ to check whether the norm condition, deadline and obligation or prohibition descriptions hold. For example, a particular monitor may query whether the speed of objects is in excess of 70 mph, but not whether it is below 60 mph. Clearly, it is impossible to

[1] Monitoring for violations of conditional norms has slightly lower complexity than monitoring $PLTL$ formulas. Detecting violations of a conditional norm using Algorithms 1 and 2 requires time linear in ρ and constant space (since monitoring of a conditional norm requires considering only the current state and a single boolean flag det indicating whether an instance of the norm has been detached in a previous state), while evaluating a past time $PLTL$ formula ψ on a finite path requires time linear in ρ and space linear in ψ [12]. However such differences are unlikely to be significant in practice.

use m_Φ to perfectly monitor for violations of norms if Φ does not contain all the formulas required to express the norms. To characterise the relation between (imperfect) monitors and norms, we need some technical definitions, which characterise the expressive power of Φ with respect to different norms and a transition system (S, R, V, s_I).

The monitor queries $\Phi = \{\phi_1, \dots, \phi_k\}$ define a partition of the set S of states of a transition system in the following way. Two states are equivalent or indistinguishable (are in the same equivalence class in the partition) with respect to the monitor m_Φ, denoted by $s \sim_\Phi s'$, if they give the same answers to all the queries in Φ[2]. Each equivalence class $[s]_\Phi, s \in S$ in the partition defined by Φ is definable by a conjunction $\sim \phi_1 \wedge \ldots \wedge \sim \phi_k$ where $\sim \phi_i$ is ϕ_i if s answers yes to (satisfies) ϕ_i, and is $\neg\phi_i$ if s answers no to (does not satisfy) ϕ_i. We will call the $\sim \phi_1 \wedge \ldots \wedge \sim \phi_k$ the 'types definable by the monitor m_Φ' or just 'types of m_Φ', $\mathcal{T}(m)$. Note that a monitor m_Φ can partition a set S of states into at most $2^{|\Phi|}$ types because it is possible that there are no states in S of a type defined by m_Φ.

We lift the partition of states of a transition system to a partition of paths. Let Γ be the set of all possible finite paths of a transition system. We say that two paths $\rho, \rho' \in \Gamma$ are equivalent with respect to a monitor m_Φ, denoted as $\rho \sim_\Phi \rho'$, if the paths are state-wise equivalent with respect to Φ, i.e.,

$$\rho \sim_\Phi \rho' \text{ iff } \forall i: \ \rho[i] \sim_\Phi \rho'[i]$$

where $\rho[i]$ is the i-th state on path ρ. A monitor m_Φ thus defines a partition of the set of paths Γ. We write $[\rho]_\Phi, \rho \in \Gamma$ to denote the equivalence class containing the path ρ, and $\Gamma_{/\sim\Phi}$ for the partition of Γ (the set of equivalence classes).

Coming back to our running example in Figure 1, suppose we have a monitor with the set of queries: $\Phi = \{p_1, p_2, p_3, s_{60} \vee s_{70}\}$ so that the monitor can identify the car's position and whether its speed is strictly above 70 mph. These queries specify four equivalence classes on the states: $\{\{s_1\}, \{s_5\}, \{s_2, s_3\}, \{s_4\}\}$. These equivalence classes are respectively characterised by the monitor types $(p_1 \wedge \neg p_2 \wedge \neg p_3 \wedge \neg s_{60} \wedge \neg s_{70} \wedge \neg s_{80})$, $(\neg p_1 \wedge \neg p_2 \wedge p_3 \wedge \neg s_{60} \wedge \neg s_{70} \wedge \neg s_{80})$, $(\neg p_1 \wedge p_2 \wedge \neg p_3 \wedge s_{60} \wedge s_{70} \wedge \neg s_{80})$, and $(\neg p_1 \wedge p_2 \wedge \neg p_3 \wedge \neg s_{60} \wedge \neg s_{70} \wedge s_{80})$. Note that the equivalence classes characterised by all other monitor types are empty given a proper domain theory (which states that the car is always in exactly one position, has at most one speed, etc.). This monitor cannot distinguish, for example, the paths $\rho_1 = s_1, s_2, s_5$ and $\rho_2 = s_1, s_3, s_5$: $\rho_1 \sim_\Phi \rho_2$.

Given a transition system M, a conditional norm n with $PLTL$ violation formula $v(n)$, a monitor m_Φ, and a path $\rho \in \Gamma$, we say that m_Φ observes ρ obeying n, denoted as $obey(m_\Phi, \rho, n)$, if $v(n)$ is false in all states on all paths in $[\rho]_\Phi$, i.e.,

$$obey(m_\Phi, \rho, n) \ \Leftrightarrow \ \forall \rho' \in [\rho]_\Phi \ \forall i : M, \rho', i \not\models v(n)$$

Note that if we have $obey(m_\Phi, \rho, n)$, then we also have $\forall \rho' \in [\rho]_\Phi : obey(m_\Phi, \rho', n)$. In other words, the equivalence class $[\rho]_\Phi$ consists of paths obeying conditional norm n. The set of equivalence classes that contain paths obeying n is denoted by $O(m_\Phi, n) = \{[\rho]_\Phi \mid obey(m_\Phi, \rho, n)\}$.

We say that the monitor m_Φ observes the path ρ violating the conditional norm n, denoted as $viol(m_\Phi, \rho, n)$, if $v(n)$ is true in some state on all paths in $[\rho]_\Phi$, i.e.,

$$viol(m_\Phi, \rho, n) \ \Leftrightarrow \ \forall \rho' \in [\rho]_\Phi \ \exists i : M, \rho', i \models v(n)$$

The set of equivalence classes that contain paths violating n is denoted by $V(m_\Phi, n) = \{[\rho]_\Phi \mid viol(m_\Phi, \rho, n)\}$.

Finally, we say that the monitor m_Φ cannot observe whether ρ violates or obeys the conditional norm n, denoted as $viob(m_\Phi, \rho, n)$, and define it as follows:

$$\begin{aligned} viob(m_\Phi, \rho, n) \ \Leftrightarrow \quad & \exists \rho' \in [\rho]_\Phi \ \forall i : M, \rho', i \not\models v(n) \ \& \\ & \exists \rho' \in [\rho]_\Phi \ \exists i : M, \rho', i \models v(n) \end{aligned}$$

The set of equivalence classes that contain paths some obeying and some violating n is denoted by $OV(m_\Phi, n) = \Gamma_{/\sim\Phi} \setminus (V(m_\Phi, n) \cup O(m_\Phi, n))$.

Definition 4 (monitor behaviour). *Given a set of monitor queries Φ, the monitor function $m_\Phi(\rho, n)$ is defined as follows:*

$$m_\Phi(\rho, n) = \begin{cases} \mathit{true} & \text{if } viol(m_\Phi, \rho, n) \\ \mathit{false} & \text{if } obey(m_\Phi, \rho, n) \\ \mathit{undefined} & \text{if } viob(m_\Phi, \rho, n) \end{cases}$$

Note that a perfect monitor never returns *undefined*.

Coming back to the running example, the transition system M_t shown in Figure 1 generates paths $(s_1(s_2|s_3|s_4)s_5)^*$. For the obligation $n_2 = (p_1, O(p_2 \wedge s_{60}), p_3)$ we have $O(m_\Phi, n_2) = \emptyset$, $V(m_\Phi, n_2) = \{[\rho]_\Phi \mid \exists i \ M_t, \rho, i \models s_{80}\}$, and $OV(m_\Phi, n_2) = \{[\rho]_\Phi \mid \exists i \ M_t, \rho, i \models s_{60} \vee s_{70}\}$ contains all partitions including paths some obeying and some violating n_2. This means that m_Φ is not a perfect monitor for n_2 on M_t.

Definition 5 (Perfect/Imperfect Monitors). *Let Γ be the set of all possible paths of a transition system, n be a conditional norm, and m_Φ be a monitor defining a partition of Γ. We say that m_Φ is a* perfect monitor *for n if and only if $OV(m_\Phi, n) = \emptyset$, and m_Φ is an* imperfect monitor *for n if and only if $OV(m_\Phi, n) \neq \emptyset$.*

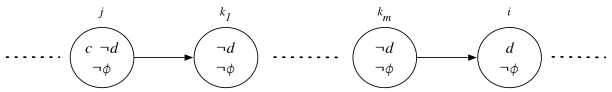
In other words, a perfect monitor for a conditional norm is able to identify all paths as either obeying or violating the norm while an imperfect monitor is unable to identify some paths as obeying or violating the norm. It is important to emphasise that the distinction between perfect and imperfect monitors is a relative distinction which depends on the set of paths and the norm under consideration. A monitor can be imperfect for a set of paths and a norm while it can be perfect for a subset of the paths (or a different set of paths) and the norm. It is however clear that if Φ contains all the queries which are needed to define formulas c, ϕ and d in a norm $n = (c, Z(\phi), d)$, then m_Φ is a perfect monitor for n:

Proposition 1. *Let $n = (c, Z(\phi), d)$ be a norm, and Φ a set of queries such that there exist a boolean combination of queries in Φ which can be used to equivalently express c, ϕ and d. Then m_Φ is a perfect monitor for n.*

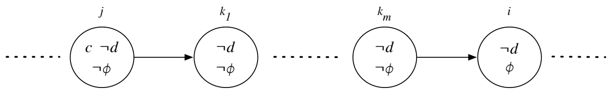
Proof. If c, ϕ and d can be equivalently rewritten in terms of queries in Φ, then $s \sim_\Phi s'$ implies that: $s \models c$ iff $s' \models c$, $s \models \phi$ iff $s' \models \phi$, and $s \models d$ iff $s' \models d$.

Suppose a path ρ violates n: there is an index i such that $\rho, i \models v(n)$. This means that there is a pattern on ρ which witnesses the violation, for example for an obligation it would be:

[2] Indistinguishability by queries in Φ is clearly a special case of an epistemic indistinguishability relation. The monitor m_Φ is guaranteed to 'know whether ψ' (i.e., $K_{m_\Phi}\psi \vee K_{m_\Phi}\neg\psi$) if ψ is definable using formulas in Φ, and is not guaranteed to know whether ψ otherwise.



and for a prohibition



For every ρ' such that $\rho \sim_\Phi \rho'$, the same pattern will exist on ρ', since $\rho[j] \sim_\Phi \rho'[j]$, $\rho[i] \sim_\Phi \rho'[i]$ and for every k in between, $\rho[k] \sim_\Phi \rho'[k]$. Hence, $\rho', i \models v(n)$. So, $OV(m_\Phi, n)$ is empty: each equivalence class either contains only the paths violating the norm, or the paths which do not. □

The distinction between perfect and imperfect monitors can be refined by introducing the notion *monitor sensitivity*. Until now we have assumed a set of state formulae $\Phi = \{\phi_1, \ldots, \phi_n\}$, which uniquely specifies a monitor. This assumed set of state formulae determines the sensitivity of the monitor with respect to a set of paths and a norm as it defines the monitor's ability to identify paths as obeying or violating the given norm. When the monitor m_Φ is perfect for a set of paths Γ and a norm n (i.e., $\Gamma_{/\sim_\Phi} \setminus (V(m_\Phi, n) \cup O(m_\Phi, n)) = \emptyset$), we say that m_Φ is fully sensitive for Γ and n. However, when the monitor is imperfect, there exist partitions that contain paths some of which obey and some others violate n.

Extending Φ with additional state formulae may make a monitor more sensitive in the sense that the set of $viob$ paths shrinks while restricting Φ may make the monitor less sensitive in the sense that the set of $viob$ paths may grow.

PROPOSITION 2. *Let Φ_1 and Φ_2 be two sets of state formulae, Γ be a set of paths, n be a conditional norm, and m_{Φ_1} and m_{Φ_2} be two monitors defining two partitions of Γ. Then, we have*

$$\Phi_1 \subseteq \Phi_2 \Leftrightarrow \bigcup OV(m_{\Phi_2}, n) \subseteq \bigcup OV(m_{\Phi_1}, n)$$

PROOF. Straightforward. □

Adding extra queries certainly allows us to implement better monitors for norms. However, sometimes providing extra queries may be too costly or impossible. A complementary approach is, given a set of queries N and a fixed set of queries Φ, come up with a set of approximations of N for which m_Φ is a perfect monitor. Intuitively, this corresponds to asking, if we cannot enforce the norms N, which norms can we enforce? This is the topic of the next section.

5. APPROXIMATING NORMS

As we have seen in the previous section, m_Φ is a perfect monitor for a norm n with condition c, obligation or prohibition ϕ, and deadline d if c, ϕ and d can be defined in terms of formulas in Φ. This, however, is not always the case. Sometimes the designer of a normative system has a given set of possible queries and has to modify the norms so that a (perfect) monitor can be implemented for the new norms, while minimising the difference in terms of missed violations between the old and the new norms.

Given a set of queries Φ and a set of 'ideal' norms N, the *norm approximation* task is to produce a set of norms N' for which m_Φ is a perfect monitor and which is the best approximation for N. By 'best' we mean that the set of violations of the approximated norm is included in the set of violations of the original norm, and the difference between the two sets is minimal.

Let us denote the language in which the conditions, deadlines, obligations and prohibitions of N are formulated L_N. We assume that we are given a propositional theory T in the language of Φ and L_N which contains some axioms relating Φ and L_N. For example, T may contain a statement that $s_{80} \rightarrow \neg(s_{60} \vee s_{70})$ where $s_{80} \in L_N$ and $s_{60} \vee s_{70} \in \Phi$ which says that the speed of 80 mph is not a speed of 60 or 70 mph. T may also be empty, in which case we only have classical propositional logic to reason about the relationship of queries definable using Φ and L_N. For example, if $p \in L_N$ and $p \vee q \in \Phi$ (but $p \notin \Phi$) we still know that $p \rightarrow p \vee q$.

In what follows, we assume that all transition systems M are models of T (that is, axioms of T are true in all states of M).

The set of paths violating n is defined relative to some transition system M.

DEFINITION 6 (VIOLATIONS). *Given a system $M = (S, R, V, s_I)$ and a norm n, a set of violations of n on M $Viol(M, n)$ is*

$$\{\rho \mid \exists i\, M, \rho, i \models v(n)\}$$

A norm n can be approximated (for detecting violations) by weakening n to a norm n' such that all paths violating n' also violate n. This means that each violation detected for n' is a violation of n, but if we monitor for violations of n', we may miss some violations of n; that is, we get some false negatives for violations of n, but never false positives. This kind of approximation makes sense in a context where agents are penalised for violating norms, and if we do not have a perfect monitor for n, we want to weaken n so we have a perfect monitor for its approximation. This way, agents will never be 'punished unfairly'.

DEFINITION 7 (APPROXIMATION). *A conditional norm n' is a* approximation *of n with respect to Φ and a background theory T (a (Φ, T)-approximation of n) iff n' is formulated in terms of the queries in Φ and the set of violations of n' is a subset of the set of violations of n in all possible models M of T:*

$$\forall M(\mathit{Viol}(M, n') \subseteq \mathit{Viol}(M, n))$$

DEFINITION 8 (OPTIMAL APPROXIMATION). *A norm n' is* the optimal approximation *of n with respect to Φ and a background theory T (an optimal (Φ, T)-approximation of n) iff it is a (Φ, T)-approximation of n and for all other (Φ, T)-approximations n'' of n,*

$$\forall M(\mathit{Viol}(M, n'') \subseteq \mathit{Viol}(M, n'))$$

The reason we define approximation (and optimality) with respect to all possible models and not with respect to a single model is because the latter notion is not robust. An optimal approximation with respect to a specific M may be exploiting some 'accidental' features of M which may become true if the environment model is slightly adjusted or the agents are given one extra possible transition which changes the set of paths. For example, it is possible that in M on every path where some norm n is detached, the state before the condition c of the norm becomes true satisfies some $p \in \Phi$ which has no connection with c in theory T. We can obtain an optimal approximation for n on M by replacing c with p. Clearly, this approximation may break down if it becomes possible to reach a c state without going through a p state.

Next we state how to define an optimal approximation with respect to Φ and T of a set of norms N which is stated in the language L_N. We assume L_N and Φ are finite.

We can construct all possible state descriptions given T, which is a finite (although in the worst case exponential in $|\Phi| + |L_N|$) set. Formally, a state description α is a formula which is consistent with respect to T and is a conjunction of all possible formulas in Φ and L_N or their negations. Let us denote the set of all possible state descriptions by $S(\Phi, L_N, T)$. For each formula ϕ in the combined language $L_N \cup \Phi$, we can define a function $val(\phi)$ which maps Φ to the state descriptions in $S(\Phi, L_N, T)$ where ϕ is true:

$$val(\phi) = \{\alpha \in S(\Phi, L_N, T) \mid \alpha \models \phi\}$$

DEFINITION 9 ((BEST) INNER APPROXIMATION). *For two propositional formulas ϕ and ϕ' in the language of $L_N \cup \Phi$, we will say that ϕ' is an* inner approximation *of ϕ with respect to Φ and T ((Φ, T)- inner approximation of ϕ) iff ϕ is written using boolean combinations of queries in Φ and $val(\phi') \subseteq val(\phi)$.*

ϕ' is the best inner approximation *of ϕ with respect to Φ and T (best (Φ, T)-inner approximation of ϕ) iff ϕ' is a (Φ, T)-inner approximation of ϕ and for every other (Φ, T)-inner approximation of ϕ, ϕ'', $val(\phi'') \subseteq val(\phi')$.*

DEFINITION 10 ((BEST) OUTER APPROXIMATION). *For two propositional formulas ϕ and ϕ' in the language of $L_N \cup \Phi$, we will say that ϕ' is an* outer approximation *of ϕ with respect to Φ and T ((Φ, T)- outer approximation of ϕ) iff ϕ is written using boolean combinations of queries in Φ and*

$$val(\phi) \subseteq val(\phi')$$

ϕ' is the best outer approximation *of ϕ with respect to Φ and T (best (Φ, T)-outer approximation of ϕ) iff ϕ' is a (Φ, T)-outer approximation of ϕ and for every other (Φ, T)-outer approximation of ϕ, ϕ'',*

$$val(\phi') \subseteq val(\phi'')$$

Next we show how to construct best inner and outer (Φ, T)-approximations of propositional formulas.

Observe that any formula ϕ' which is definable in terms of queries from Φ can be equivalently written as $\bigvee \theta_i$, where each $\theta_i \in \mathcal{T}(m_\phi)$, that is, is of the form

$$(\sim \phi_1 \wedge \ldots \wedge \sim \phi_k)$$

where $\sim \phi_j$ is either ϕ_j or its negation. For each of the $2^{|\Phi|}$ possible θ_i, we compute the subset $val(\theta_i)$ of $S(\Phi, L_N, T)$ where θ_i is true. We define

$$\phi^-_{(\Phi,T)} = \bigvee_{\{\theta_i \in \mathcal{T}(m_\Phi) \mid val(\theta_i) \subseteq val(\phi)\}} \theta_i$$

$$\phi^+_{(\Phi,T)} = \bigvee_{\{\theta_i \in \mathcal{T}(m_\Phi) \mid val(\theta_i) \cap val(\phi) \neq \emptyset\}} \theta_i$$

In what follows, we will omit the subscript (Φ, T) in $\phi^-_{(\Phi,T)}$ and $\phi^+_{(\Phi,T)}$ for readability.

Clearly, $T \models \phi^- \rightarrow \phi$ and $T \models \phi \rightarrow \phi^+$.

Coming back to the running example in Figure 1, observe that the language L_N contains $\{p_1, p_2, p_3, s_{60}, s_{70}, s_{80}\}$, and $\Phi = \{p_1, p_2, p_3, s_{60} \vee s_{70}\}$. The domain theory T will contain statements about the speed of the car and position of the car being unique, and that the car always has some position, etc. The set $S(\Phi, L_N, T)$ of all possible state descriptions contains, for example, a description of s_4: $\alpha_4 = \neg p_1 \wedge p_2 \wedge \neg p_3 \wedge \neg s_{60} \wedge \neg s_{70} \wedge s_{80}$, similarly for s_2 and s_3. Note that the set of monitor types or state descriptions in terms of Φ is smaller. It only contains two rather than three possibilities for the car being in position p_2:

$$\theta_1 = \neg p_1 \wedge p_2 \wedge \neg p_3 \wedge (s_{60} \vee s_{70})$$

$$\theta_2 = \neg p_1 \wedge p_2 \wedge \neg p_3 \wedge \neg(s_{60} \vee s_{70})$$

For the formula $\phi = (p_2 \wedge s_{60})$, $\phi^- = \bot$ and $\phi^+ = \theta_2$.

Before we prove that ϕ^- and ϕ^+ are the best inner and outer approximations of ϕ, we need a lemma which will also be useful in later proofs.

LEMMA 1. *If a state description $\alpha \in S(\Phi, L_N, T)$ satisfies a formula ϕ' written using boolean combinations of formulas from Φ, and does not satisfy ϕ^-, then ϕ' is not a (Φ, T)-inner approximation of ϕ.*

If a state description $\alpha \in S(\Phi, L_N, T)$ satisfies ϕ^+ and does not satisfy a formula ϕ' written using boolean combinations of formulas from Φ, then ϕ' is not a (Φ, T)-outer approximation of ϕ.

PROOF. Straightforward. □

PROPOSITION 3. *$\phi^-_{(\Phi,T)}$ and $\phi^+_{(\Phi,T)}$ are the best inner and outer (Φ, T)-approximations of ϕ.*

PROOF. Clearly, $val(\phi^-) \subseteq val(\phi)$ and $val(\phi) \subseteq val(\phi^+)$. To show that ϕ^- is the best inner approximation of ϕ, assume that there is another (Φ, T)-inner approximation ϕ' of ϕ and a state description $\alpha \in val(\phi)$ such that $\alpha \in val(\phi')$ and $\alpha \notin val(\phi^-)$. By Lemma 1, ϕ' is not an inner approximation of ϕ.

The proof that ϕ^+ is the best outer approximation of ϕ is similar. □

Next we state the method to produce the optimal (Φ, T)-approximations for violations given the original set of norms, and a theorem which states that any other approximation will miss at least as many violations as the one computed by the method.

Given an obligation $(c, O(\phi), d)$, we need to produce an obligation $(c', O(\phi'), d')$ where c', ϕ', d' are definable in terms of queries from Φ. We do this as follows:

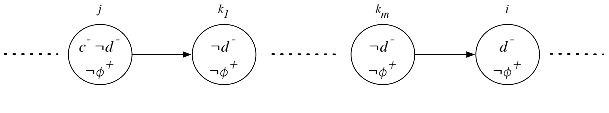
- $c' = c^-$ (c' is the best inner approximation of c, fewer states satisfy c' than c)
- $\phi' = \phi^+$ (ϕ' is the best outer approximation of ϕ, more states satisfy the obligation description ϕ')
- $d' = d^-$ (d' is the best inner approximation of d, fewer states satisfy d')

These conditions are intended to guarantee that the approximation of an obligation is violated only if the original obligation is violated:

LEMMA 2. *For any conditional obligation $n = (c, O(\phi), d)$, its approximation $n' = (c^-, O(\phi^+), d^-)$, model M, path ρ in M, if $\exists i\, M, \rho, i \models v(n')$, then $\exists i\, M, \rho, i \models v(n)$.*

PROOF. Assume $M, \rho, i \models v(n')$, that is, $M, \rho, i \models d^- \wedge \neg\phi^+ \wedge ((\mathcal{Y}\,(\neg\phi^+ \wedge \neg d^-)\,\mathcal{S}\,(c^- \wedge \neg\phi^+ \wedge \neg d^-)) \vee c^-)$.

This means that there is a pattern on ρ:



where possibly $j = i$ (in which case $m = 0$).

Clearly, for any M, ρ, i where M is a model of T, if $M, \rho, i \models d^-$, then $M, \rho, i \models d$ (the definition of inner approximation with respect to T). Similarly, for every M, ρ, i, if $M, \rho, i \models c^-$, then $M, \rho, i \models c$. Finally, by the definition of the outer approximation, $M, \rho, i \models \neg\phi^+$, then $M, \rho, i \models \neg\phi$.

So, if $M, \rho, j \models c^- \land \neg\phi^+$, then also $M, \rho, j \models c \land \neg\phi$, and if $M, \rho, i \models d^- \land \neg\phi^+$, then also $M, \rho, i \models d \land \neg\phi$. However $M, \rho, k \models \neg\phi^+ \land \neg d^-$ does not imply $M, \rho, k \models \neg\phi \land \neg d$ (because $\neg d^-$ is easier to satisfy). Basically, a violation of n may occur in a different state on ρ (before i), however if there is a violation of n' (some index k satisfies $\neg\phi^+$) then there is a violation of n, so n' is an approximation. □

For a prohibition $(c, P(\phi), d)$, where c, ϕ, we need to produce a prohibition $(c', P(\phi'), d')$ where c', ϕ', d' are definable in terms of queries from Φ. We define the approximation as $(c^-, P(\phi^-), d^+)$. This definition is intended to guarantee that the approximation of a prohibition is violated only if the original prohibition is violated:

LEMMA 3. *For any conditional prohibition $n = (c, P(\phi), d)$, its approximation $n' = (c^-, P(\phi^-), d^+)$, model M, path ρ in M, if $\exists i\ M, \rho, i \models v(n')$, then $\exists i\ M, \rho, i \models v(n)$.*

PROOF. The proof is similar to the proof of Lemma 2. □

Now we can prove correctness of our construction of the set of optimal norm approximations.

THEOREM 3. *Given N and Φ, the following set N' is the set of optimal approximations of the norms in N:*

$$\begin{aligned} N' &= \{(c^-, O(\phi^+), d^-) \mid (c, O(\phi), d) \in N\} \cup \\ &\quad \{(c^-, P(\phi^-), d^+) \mid (c, P(\phi), d) \in N\} \end{aligned}$$

PROOF. By Proposition 1, norms in N' can be perfectly monitored by m_Φ since they are defined in terms of Φ. We need to show for every $n \in N$, that its approximation n' is an approximation of n for violations, formally on every M,

$$Viol(M, n') \subseteq Viol(M, n)$$

and that it is optimal, i.e., for every other approximation n'',

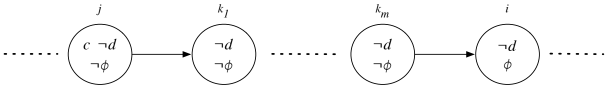
$$\forall M (Viol(M, n'') \subseteq Viol(M, n'))$$

To prove that n' is an approximation of n for violations, suppose that for some M, $\rho \in Viol(M, n')$, that is, for some i, $M, \rho, i \models v(n')$. From Lemmas 2 and 3 it follows that for every M, ρ if for some i, $M, \rho, i \models v(n')$ then for some j, $M, \rho, j \models v(n)$. This means that $\rho \in Viol(M, n)$. So, for every M, $Viol(M, n') \subseteq Viol(M, n)$.

To prove that n' is an optimal approximation of n for violations, suppose by contradiction that for some other norm n'' definable in terms of Φ which is an approximation of n, there is some model M of T and a path ρ such that:

1. for some i, $M, \rho, i \models v(n)$ (there is a violation of n)
2. for some i', $M, \rho, i' \models v(n'')$ (it is also a violation of n'')
3. for all i, $M, \rho, i \not\models v(n')$ (it is not a violation of n', so monitoring for n' this violation of n will be missed)

Let us consider the case when n is a prohibition, $n = (c, P(\phi), d)$. Since n is violated on some ρ, there exists a pattern of states on ρ



Since n'' is violated on ρ, there exists a similar pattern of states on ρ for c'' and ϕ''. Let us call the state where c'' is true s_1, and the state where ϕ is true s_2. Note that it is possible that $s_1 = s_2$. Observe also that either $s_1 \not\models c'$, or $s_2 \not\models \phi'$, or some state between s_1 and s_2 on ρ satisfies d' but not d''. It can be shown in all these cases that then we can construct a model M' and a path ρ' in M' such that ρ' contains a violation of n'' but not of n, so n'' is not an approximation of n.

Consider the case when $s_1 \not\models c'$ and $s_1 \neq s_2$. By Lemma 1, c'' is not an inner approximation of c, hence there exists a state description α such that $\alpha \in val(c'')$ and $\alpha \notin val(c)$. Consider a state s' which conforms to this description, and consider a model M' which consists of a single step path from s' to s_2. This path will be a violation of n'' since s' satisfies c'' and s_2 satisfies ϕ'', but it will not be a violation of n since n is not detached in s'. Let us consider the case when $s_1 = s_2$. Suppose by contradiction that it is impossible to construct a state s' satisfying c'' and and ϕ'' (a violation of n'') but not satisfying c or ϕ (hence not a violation of n). This would only be the case if $T \models (c'' \land \phi'') \equiv (c \land \phi)$ and $T \models (c \equiv \phi)$. The latter entails $T \models (c' \equiv \phi')$. Since we know that $s_1 = s_2$ does not violate n', c' is not equivalent to c'' and ϕ''. Since c' and c'' are in the language of Φ, there is a θ' such that $val(\theta') \subseteq val(c'')$ such that $val(\theta') \not\subseteq val(c)$, so there exists a state which satisfies c'' but does not satisfy c and ϕ. Other cases for a conditional prohibition (when $s_2 \not\models \phi'$ or one of the intermediate states satisfies d') are similar.

The case for obligations is analogous. □

The stated method of finding an optimal norm approximation is computationally expensive (exponential in the size of the original set of norms). However we can show that it is still optimal since the problem of norm approximation itself is computationally hard.

Since we know from Theorem 3 that components of conditional norms have to be weakened in a particular way to obtain an optimal approximation, the norm approximation problem can be re-stated as follows:

DEFINITION 11. Norm approximation problem:

Input: *A theory T (which may be empty), a set of queries Φ and a norm $(c, Z(\phi), d)$ where $Z \in \{O, P\}$*

Output: *For each formula $\psi \in \{c, \phi, d\}$, find ψ_1 and ψ_2 such that:*

- $T \models \psi_1 \to \psi$
- *for all ψ' in the language of Φ such that $T \models \psi' \to \psi$, it holds that $T \models (\psi' \to \psi_1)$*
- $T \models \psi \to \psi_2$
- *for all ψ' in the language of Φ such that $T \models \psi \to \psi'$, it holds that $T \models \psi_2 \to \psi'$.*

To establish the complexity of the norm approximation problem, we compare it to the following *interpolation problem*:

DEFINITION 12. Interpolation problem:

Input: *A classical propositional tautology $\phi \to \psi$*

Output: *A formula χ (an interpolant, [8]) in the common language of ϕ and ψ such that $\phi \to \chi$ and $\chi \to \psi$ are tautologies.*

The lower bound for the size of the interpolant is exponential in the size of $\phi \to \psi$, if there is a set in $NP \cap coNP$ that cannot be computed by subexponential size circuits [15, 13]. The latter hypothesis is a standard assumption in cryptography. So for all practical purposes, some tautologies only have exponential size interpolants, and hence the interpolation problem above also has EXPTIME lower bound.

THEOREM 4 (COMPLEXITY OF NORM APPROXIMATION). *The norm approximation problem is at least as hard as the interpolation problem.*

PROOF. We give a polynomial reduction from the interpolation problem to the norm approximation problem. Take a tautology $\phi \rightarrow \psi$. Make two norms, $(\top, O(\phi), \bot)$ and $(\top, P(\psi), \bot)$, and set Φ to be the set of propositional variables in the common language of ϕ and ψ. Then we can prove that the interpolant is ϕ_2 defined in Definition 11, hence a solution to norm approximation problem gives us a solution to the interpolation problem.

By the definition of norm approximation problem, $\models \phi \rightarrow \phi_2$. There is (by the Craig interpolation theorem [8]) an interpolant ρ in the same language as ϕ_2 such that $\phi \rightarrow \rho$ and $\rho \rightarrow \psi$. Since ϕ_2 is the strongest formula for which $\phi \rightarrow \phi_2$ holds, $\phi_2 \rightarrow \rho$. Hence $\phi_2 \rightarrow \psi$. □

6. DISCUSSION

There is a growing body of literature on logical analysis of norms in multi-agent systems, for example [1, 9]. Most approaches to normative multi-agent organisation assume perfect monitoring mechanisms, i.e., that monitors are able to detect all violations or compliances of a given norm (see e.g., [10, 14]). An exception to these perfect monitor approaches is the recent work by Bulling et al. [7].

Our approach is similar to [7] in that we also define a monitor in terms of (in)distinguishable execution traces. However, in contrast to [7], we define a monitor based on indistinguishable states and then lift it to an indistinguishability relation on execution traces. Moreover, while [7] investigates the problem of combining monitors to construct ideal monitors, we investigate how norms can be modified such that a given imperfect monitor can function as a perfect monitor for a norm approximation. Another important distinction is our use of conditional norms with deadlines evaluated on finite execution traces rather than using arbitrary LTL formulae as norms evaluated on infinite traces, as proposed in [7]. Although the latter approach allows more expressive norms and monitors, our approach has more potential applications. Typical applications are systems that require runtime monitoring, for example, monitoring web services, policy monitoring, debugging, fault monitoring, and runtime repair and recovery of system executions (see [4, 5, 11]).

Our approach is also related to runtime verification of computer programs, where the current execution of a system is monitored and evaluated with respect to some desirable properties [6]. Runtime verification is often used to detect the violation of correctness properties expressed in some temporal language such as LTL. Like runtime verification our approach is based on finite execution traces, but unlike these approaches we focus on approximation of properties that can be observed and evaluated by an imperfect monitor. Our approach is also tailored to normative multi-agent organisations, where conditional norms with deadlines are used to coordinate the (inter)action of individual agents.

7. REFERENCES

[1] T. Ågotnes, W. van der Hoek, and M. Wooldridge. Robust normative systems and a logic of norm compliance. *Logic Journal of the IGPL*, 18(1):4–30, 2010.

[2] N. Alechina, M. Dastani, and B. Logan. Programming norm-aware agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, pages 1057–1064. IFAAMAS, 2012.

[3] N. Alechina, M. Dastani, and B. Logan. Reasoning about normative update. In *Proceedings of the Twenty Third International Joint Conference on Artificial Intelligence (IJCAI 2013)*, pages 20–26. AAAI Press, 2013.

[4] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti. Run-time monitoring of instances and classes of web service compositions. *19th International Conference on Web Services (ICWS'06)*, pages 63–71, 2006.

[5] L. Baresi, S. Guinea, and L. Pasquale. Towards a unified framework for the monitoring and recovery of BPEL processes. In T. Bultan and T. Xie, editors, *Workshop on Testing, Analysis, and Verification of Web Services and Applications (TAV-WEB)*, pages 15–19. ACM, 2008.

[6] A. Bauer, M. Leucker, and C. Schallhart. Runtime verification for LTL and TLTL. *ACM Transactions on Software Engineering and Methodology*, 20(4):1–68, 2011.

[7] N. Bulling, M. Dastani, and M. Knobbout. Monitoring norm violations in multi-agent systems. In *Twelfth International conference on Autonomous Agents and Multi-Agent Systems (AAMAS'13)*, pages 491–498, 2013.

[8] W. Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *J. Symb. Log.*, 22(3):269–285, 1957.

[9] M. Dastani, D. Grossi, J.-J. C. Meyer, and N. Tinnemeier. Normative multi-agent programs and their logics. In *Proc. Workshop on Knowledge Representation for Agents and Multi-Agent Systems*, LNCS 5605, pages 16–31, 2009.

[10] M. Dastani, J.-J. C. Meyer, and D. Grossi. A logic for normative multi-agent programs. *Journal of Logic and Computation*, 23(2):335–354, 2013.

[11] N. Delgado, A. Q. Gates, and S. Roach. A taxonomy and catalog of runtime software-fault monitoring tools. *IEEE Transactions on Software Engineering*, 30(12):859–872, 2004.

[12] K. Havelund and G. Rosu. Synthesizing monitors for safety properties. In J.-P. Katoen and P. Stevens, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 8th International Conference, TACAS 2002*, volume 2280 of *LNCS*, pages 342–356. Springer, 2002.

[13] J. Krajícek. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *J. Symb. Log.*, 62(2):457–486, 1997.

[14] S. Modgil, N. Faci, F. R. Meneguzzi, N. Oren, S. Miles, and M. Luck. A framework for monitoring agent-based normative systems. In *Eighth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'09)*, pages 153–160, 2009.

[15] D. Mundici. A lower bound for the complexity of Craig's interpolants in sentential logic. *Arch. Math. Logic*, 23:27–36, 1983.

[16] N. Tinnemeier, M. Dastani, J.-J. Meyer, and L. van der Torre. Programming normative artifacts with declarative obligations and prohibitions. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'09)*, pages 69–78, 2009.