

Robust Normative Systems

Thomas Ågotnes
Dept of Computer Engineering
Bergen University College
PB. 2030, N-5020 Bergen
Norway
tag@hib.no

Wiebe van der Hoek
Dept of Computer Science
University of Liverpool
Liverpool L69 7ZF
UK
wiebe@csc.liv.ac.uk

Michael Wooldridge
Dept of Computer Science
University of Liverpool
Liverpool L69 7ZF
UK
mjw@liv.ac.uk

ABSTRACT

Although normative systems, or social laws, have proved to be a highly influential approach to coordination in multi-agent systems, the issue of *compliance* to such normative systems remains problematic. In all real systems, it is possible that some members of an agent population will not comply with the rules of a normative system, even if it is in their interests to do so. It is therefore important to consider the extent to which a normative system is *robust*, i.e., the extent to which it remains effective even if some agents do not comply with it. We formalise and investigate three different notions of robustness and related decision problems. We begin by considering sets of agents whose compliance is necessary and/or sufficient to guarantee the effectiveness of a normative system; we then consider quantitative approaches to robustness, where we try to identify the proportion of an agent population that must comply in order to ensure success, and finally, we consider a more general approach, where we characterise the compliance conditions required for success as a logical formula.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multiagent Systems;
I.2.4 [Knowledge representation formalisms and methods]

General Terms

Theory

Keywords

normative systems, robustness, fault tolerance, complexity

1. INTRODUCTION

Normative systems, or social laws, have been widely promoted as an approach to coordinating multi-agent systems [11, 12, 6, 8, 1, 2]. The basic idea is that a normative system is a set of constraints on the behaviour of agents in the system; after imposing these constraints, it is intended that some desirable overall property will hold. One of the most important issues associated with such normative systems – and one of the most ignored – is that of *compliance*. Put simply, what happens if some system participants do not comply with the regulations of the normative system? Non-compliance may be accidental (e.g., a message fails and so some participants

are not informed about the regulations). Alternatively, it may be deliberate but rational (e.g., a participant chooses to ignore the norms because it does not see them as being in its own best interests), or deliberately irrational (e.g., a computer virus). Whatever the cause, it seems inevitable that, in real, large-scale systems, non-compliance will occur, and it is therefore important to consider the consequences of non-compliance. Existing research has addressed the issue of non-compliance in at least two ways.

First, one can design the normative system taking the goals and aspirations of system participants into account, so that compliance is the rational choice for participants [2]. Using the terminology of mechanism design [10, p.179], we try to make compliance *incentive compatible*. Where this approach is available, it seems highly attractive. However, given some desired objective for a normative system, it is not always possible to construct an incentive compatible normative system that achieves some outcome, and even where it is possible, it is still likely that large, open systems will fall prey to irrational behaviour.

Second, one can combine the normative system with some *penalty* mechanism, to punish non-compliance [4]. The advantage of this approach is that it can be applied to most scenarios, and that it is familiar (this is, after all, how normative systems often work in the real world). There are many disadvantages, however. For example, it may be hard to detect when non-compliance has occurred, and in large, Internet-like systems, it may be hard to impose penalties (e.g., across national borders).

For these reasons, in this paper we introduce the notion of *robustness* for normative systems. Intuitively, a normative system is robust to the extent to which it remains effective in the event of non-compliance by some agents. Following an introduction to the technical framework of normative systems, we introduce and investigate three ways of characterising robustness. First, we consider trying to identify coalitions whose compliance is *necessary* and/or *sufficient* to ensure that the normative system is effective. We characterise the complexity of checking these notions of robustness, and consider cases where verifying these notions of robustness is easier. In addition to verification we consider the complexity of *robust feasibility* of a normative system: given a reliable coalition, does there exist a normative system which is effective whenever that coalition complies? We then consider a more *quantitative* notion of robustness, called *k-robustness*, where we try to identify the *number* of agents that could deviate and still leave the normative system effective. Finally, we consider a more general, *logical* approach of characterising robustness, whereby we define a predicate over sets of agents, such that this predicate characterises exactly those sets of agents whose compliance will ensure the success of the normative system. We conclude with a brief discussion, including some pointers to related and future work.

Cite as: Robust Normative Systems, T. Ågotnes, W. van der Hoek, and M. Wooldridge, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. XXX-XXX. Copyright © 2008, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

2. FORMAL PRELIMINARIES

In this section, we present the formal framework for normative systems that we use throughout the remainder of the paper. This framework is based on that of [8, 1, 2], which is in turn descended from [11]. Although our presentation is complete, it is succinct, and readers are referred to [8, 1, 2] for details and discussion.

Kripke Structures: We use *Kripke structures* as our basic semantic model for multi-agent systems [5]. A Kripke structure is essentially a directed graph, with the vertex set S corresponding to possible *states* of the system being modelled, and the relation $R \subseteq S \times S$ capturing the possible *transitions* of the system; $S^0 \subseteq S$ denotes the *initial states* of the system. Intuitively, transitions are caused by *agents* in the system performing *actions*, although we do not include such actions in our semantic model (see, e.g., [11, 8] for models which include actions as first class citizens). An arc $(s, s') \in R$ corresponds to the execution of an atomic action by one of the agents in the system. Note that we are therefore here *not* modelling *synchronous* action. This assumption is not essential, but it simplifies the presentation. However, we find it convenient to include within our model the agents that cause transitions. We therefore assume a set A of agents, and we label each transition in R with the agent that causes the transition via a function $\alpha : R \rightarrow A$. Finally, we use a vocabulary $\Phi = \{p, q, \dots\}$ of Boolean variables to express the properties of individual states S : we use a function $V : S \rightarrow 2^\Phi$ to label each state with the Boolean variables true (or satisfied) in that state.

Formally, an *agent-labelled Kripke structure* (over Φ) is a 6-tuple:

$$K = \langle S, S^0, R, A, \alpha, V \rangle,$$

where: S is a finite, non-empty set of *states*; $S^0 \subseteq S$ ($S^0 \neq \emptyset$) is the set of *initial states*; $R \subseteq S \times S$ is a total binary relation on S , which we refer to as the *transition relation*; $A = \{1, \dots, n\}$ is a set of *agents*; $\alpha : R \rightarrow A$ labels each transition in R with an agent; and $V : S \rightarrow 2^\Phi$ labels each state with the set of propositional variables true in that state.

We hereafter refer to an agent-labelled Kripke structure simply as a *Kripke structure*. A *path* over a transition relation R is an infinite sequence of states $\pi = s_0, s_1, \dots$ such that $\forall u \in \mathbb{N}$: $(s_u, s_{u+1}) \in R$. If $u \in \mathbb{N}$, then we denote by $\pi[u]$ the component indexed by u in π (thus $\pi[0]$ denotes the first element, $\pi[1]$ the second, and so on). A path π such that $\pi[0] = s$ is an *s-path*. Let $\Pi_R(s)$ denote the set of *s-paths* over R ; since it will usually be clear from context, we often omit reference to R , and simply write $\Pi(s)$. We will sometimes refer to and think of an *s-path* as a possible computation, or system evolution, from s .

CTL: We use Computation Tree Logic (CTL), a well-known and widely used branching time temporal logic, to express the *objectives* of normative systems [5]. Given a set $\Phi = \{p, q, \dots\}$ of atomic propositions, the syntax of CTL is defined by the following grammar, where $p \in \Phi$:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \text{EO}\varphi \mid \text{E}(\varphi\mathcal{U}\varphi) \mid \text{AO}\varphi \mid \text{A}(\varphi\mathcal{U}\varphi)$$

The semantics of CTL are given with respect to the satisfaction relation “ \models ”, which holds between *pointed structures* K, s , (where K is a Kripke structure and s is a state in K), and formulae of the language. The satisfaction relation is defined as follows:

$$\begin{aligned} K, s &\models \top; \\ K, s &\models p \text{ iff } p \in V(s) \quad (\text{where } p \in \Phi); \\ K, s &\models \neg\varphi \text{ iff not } K, s \models \varphi; \end{aligned}$$

$$\begin{aligned} K, s &\models \varphi \vee \psi \text{ iff } K, s \models \varphi \text{ or } K, s \models \psi; \\ K, s &\models \text{AO}\varphi \text{ iff } \forall \pi \in \Pi(s) : K, \pi[1] \models \varphi; \\ K, s &\models \text{EO}\varphi \text{ iff } \exists \pi \in \Pi(s) : K, \pi[1] \models \varphi; \\ K, s &\models \text{A}(\varphi\mathcal{U}\psi) \text{ iff } \forall \pi \in \Pi(s), \exists u \in \mathbb{N}, \text{ s.t. } K, \pi[u] \models \psi \text{ and } \\ &\forall v, (0 \leq v < u) : K, \pi[v] \models \varphi \\ K, s &\models \text{E}(\varphi\mathcal{U}\psi) \text{ iff } \exists \pi \in \Pi(s), \exists u \in \mathbb{N}, \text{ s.t. } K, \pi[u] \models \psi \text{ and } \\ &\forall v, (0 \leq v < u) : K, \pi[v] \models \varphi \end{aligned}$$

The remaining classical logic connectives (“ \wedge ”, “ \rightarrow ”, “ \leftrightarrow ”) are defined as abbreviations in terms of \neg, \vee in the conventional way. The remaining CTL temporal operators are defined:

$$\begin{aligned} \text{A}\diamond\varphi &\equiv \text{A}(\top\mathcal{U}\varphi) & \text{E}\diamond\varphi &\equiv \text{E}(\top\mathcal{U}\varphi) \\ \text{A}\square\varphi &\equiv \neg\text{E}\diamond\neg\varphi & \text{E}\square\varphi &\equiv \neg\text{A}\diamond\neg\varphi \end{aligned}$$

We say φ is *satisfiable* if $K, s \models \varphi$ for some Kripke structure K and state s in K ; φ is *valid* if $K, s \models \varphi$ for all Kripke structures K and states s in K . The problem of checking whether $K, s \models \varphi$ for given K, s, φ (*model checking*) can be done in deterministic polynomial time, while checking whether a given φ is satisfiable or whether φ is valid is EXPTIME-complete [5]. We write $K \models \varphi$ if $K, s_0 \models \varphi$ for all $s_0 \in S^0$, and $\models \varphi$ if $K \models \varphi$ for all K .

Later, we will make use of two fragments of CTL: the universal language L^u (with typical element μ), and the existential fragment L^e (typical element ε):

$$\begin{aligned} \mu &::= \top \mid \perp \mid p \mid \neg p \mid \mu \vee \mu \mid \mu \wedge \mu \mid \text{AO}\mu \mid \text{A}\square\mu \mid \text{A}(\mu\mathcal{U}\mu) \\ \varepsilon &::= \top \mid \perp \mid p \mid \neg p \mid \varepsilon \vee \varepsilon \mid \varepsilon \wedge \varepsilon \mid \text{EO}\varepsilon \mid \text{E}\square\varepsilon \mid \text{E}(\varepsilon\mathcal{U}\varepsilon) \end{aligned}$$

The key point about these fragments is as follows. Let us say, for two Kripke structures $K_1 = \langle S, S^0, R_1, A, \alpha, V \rangle$ and $K_2 = \langle S, S^0, R_2, A, \alpha, V \rangle$ that K_1 is a subsystem of K_2 and K_2 is a supersystem of K_1 , (denoted $K_1 \sqsubseteq K_2$), iff $R_1 \subseteq R_2$. Then we have (cf. [8]).

THEOREM 1 ([8]). *Suppose $K_1 \sqsubseteq K_2$, and $s \in S$. Then:*

$$\begin{aligned} \forall \varepsilon \in L^e : K_1, s \models \varepsilon &\Rightarrow K_2, s \models \varepsilon; \quad \text{and} \\ \forall \mu \in L^u : K_2, s \models \mu &\Rightarrow K_1, s \models \mu. \end{aligned}$$

Normative Systems: For our purposes, a *normative system* (or “norm”) is simply a *set of constraints on the behaviour of agents in a system* [1]. More precisely, a normative system defines, for every possible system transition, whether or not that transition is considered to be legal or not. Different normative systems may differ on whether or not a transition is legal. Formally, a normative system η (w.r.t. a Kripke structure $K = \langle S, S^0, R, A, \alpha, V \rangle$) is simply a subset of R , such that $R \setminus \eta$ is a total relation. The requirement that $R \setminus \eta$ is total is a *reasonableness* constraint: it prevents normative systems which lead to states with no successor. Let $N(R) = \{\eta : (\eta \subseteq R) \ \& \ (R \setminus \eta \text{ is total})\}$ be the set of normative systems over R . The intended interpretation of a normative system η is that $(s, s') \in \eta$ means transition (s, s') is forbidden in the context of η . We denote the *empty* normative system by η_\emptyset , i.e., $\eta_\emptyset = \emptyset$. Let $A(\eta) = \{\alpha(s, s') \mid (s, s') \in \eta\}$ denote the set of agents involved in η .

The effect of *implementing* a normative system on a Kripke structure is to eliminate from it all transitions that are forbidden according to this normative system (see [8, 1]). If K is a Kripke structure, and η is a normative system over K , then $K \dagger \eta$ denotes the Kripke structure obtained from K by deleting transitions forbidden in η . Formally, if $K = \langle S, S^0, R, A, \alpha, V \rangle$, and $\eta \in N(R)$, then let $K \dagger \eta = K'$ be the Kripke structure $K' = \langle S', S^{0'}, R', A', \alpha', V' \rangle$ where:

- $S = S', S^0 = S'^0, A = A',$ and $V = V'$;
- $R' = R \setminus \eta$; and
- α' is the restriction of α to R' :

$$\alpha'(s, s') = \begin{cases} \alpha(s, s') & \text{if } (s, s') \in R' \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The next most basic question we can ask in the context of normative systems is as follows. We are given a Kripke structure K , representing the state transition graph of our system, and we are given a CTL formula φ , representing the *objective* of a normative system designer (that is, the objective characterises what a designer wishes to accomplish with a normative system). The *feasibility* problem is then whether or not there exists a normative system η such that implementing η in K will achieve φ , i.e., whether $K \uparrow \eta \models \varphi$. We say that η is effective for φ in K if $K \uparrow \eta \models \varphi$.

We make use of operators on normative systems which correspond to groups of agents “defecting” from the normative system. Formally, let $K = \langle S, S^0, R, A, \alpha, V \rangle$ be a Kripke structure, let $C \subseteq A$ be a set of agents over K , and let η be a normative system over K . Then $\eta \upharpoonright C$ denotes the normative system that is the same as η except that it only contains the arcs of η that correspond to the actions of agents in C , i.e., $\eta \upharpoonright C = \{(s, s') : (s, s') \in \eta \ \& \ \alpha(s, s') \in C\}$. Also, $\eta \upharpoonright \bar{C}$ denotes the normative system that is the same as η except that it only contains the arcs of η that *do not* correspond to actions of agents in C : $\eta \upharpoonright \bar{C} = \{(s, s') : (s, s') \in \eta \ \& \ \alpha(s, s') \notin C\}$.

3. NECESSITY AND SUFFICIENCY

As we noted in the introduction, the basic intuition behind robust normative systems is that they remain effective in the presence of deviation, or non-compliance, by some members of the agent population. As we shall see, there are several different ways of formulating robustness. Our first approach is to try to characterise “lynchpin” agents – those agents whose compliance with the normative system is somehow crucial for the successful operation of the system. This seems appropriate when there are “key players” in the normative system – for example, where there is a single point of failure. In this section, we therefore consider coalitions whose compliance is *necessary and/or sufficient* to ensure that the normative system is effective.

We say that $C \subseteq A$ are *sufficient* for η in the context of K and φ if the compliance of C with η is effective, i.e., iff:

$$\forall C' \subseteq A : (C \subseteq C') \Rightarrow [K \uparrow (\eta \upharpoonright C') \models \varphi].$$

The following example illustrates this notion of sufficiency.

EXAMPLE 1. Consider four agents who are attending a conference with an on-site computer facility. This service centre has currently one printer, two scanners and three PCs available. Agent a has tasks that require access to a printer and PC, agent b needs a printer and scanner, agent c is in need of a scanner and PC and agent d will need a scanner only. The set of agents is $A = \{a, b, c, d\}$. They are interested in using resources of type R_1, R_2, R_3 , of each resource type R_j there are j instances of each: $R_1 = \{\text{printer}_1\}$, $R_2 = \{\text{scanner}_1, \text{scanner}_2\}$, $R_3 = \{pc_1, pc_2, pc_3\}$. At a given point in time, a resource can be owned by an agent. The actions available to the agents are making available a resource they currently own, or taking possession of a resource which is available. We assume that the agents never act at exactly the same time; in particular we assume that actions are turn-based – first a can perform some action, then b , and so on. A state s is a tuple

$$s = \langle O_a, O_b, O_c, O_d, i \rangle$$

where, for each $i \in A$, O_i is the set of resources currently owned by i .

The number of agents that own a resource of type j cannot be greater than j . Let, for each resource R_j and state s , $\text{avail}(j, s)$ be the number of resources of type j that are not owned by an agent. The component $i \in A$ of s denotes whose turn it is: we write $\text{turn}(s) = i$. If $R_j \cap O_i \neq \emptyset$, we say that i owns a resource of type j and write $R_j \prec O_i$.

Our agents are not equal. In order to fulfil his task, agent a would every now and then like to use resources of type R_1 and R_3 simultaneously. We write $\text{Useful}(a) = \{R_1, R_3\}$. Similarly, $\text{Useful}(b) = \{R_1, R_2\}$, $\text{Useful}(c) = \{R_2, R_3\}$ while $\text{Useful}(d) = \{R_2\}$.

Let $s = \langle O_a, O_b, O_c, O_d, i \rangle$ and $s' = \langle O'_a, O'_b, O'_c, O'_d, i' \rangle$ be two states. Then $(s, s') \in R$ iff

1. $a' = b, b' = c, c' = d$ and $d' = a$;
2. for all $k \neq i$ and all j : $R_j \prec O_k \Leftrightarrow R_j \prec O'_k$;
3. if $R_j \prec O'_i$ and $R_j \not\prec O_i$ then $\text{avail}(j, s) > 0$.

Furthermore, $\alpha(s, s') = i$ when $\text{turn}(s) = i$.

Let the starting state of the system be such that it is agent a 's turn, and nobody owns any resource. If we call this system K_0 , then a first norm η_0 we impose on K is that no agent (i) owns two resources of the same type at the same time, (ii) takes possession of a resource that he does not need, (iii) takes possession of two new resources simultaneously, and (iv) fails to take possession of some useful resource if it is available when it is his turn:

$$\eta_0 = \left\{ (s, s') \mid \begin{array}{l} \text{turn}(s) = i, \text{ and} \\ (\exists j : |O'_i \cap R_j| \geq 2, \text{ or} \\ \exists j : |O'_i \cap R_j| \geq 1 \text{ and } R_j \notin \text{Useful}(i), \text{ or} \\ \exists x, y : x \neq y, x, y \in O'_i \text{ and } x, y \notin O_i, \text{ or} \\ \forall j : (R_j \in \text{Useful}(i), |O_i \cap R_j| = 0, \\ \text{avail}(j, s) > 0) \Rightarrow |O'_i \cap R_j| = 0). \end{array} \right\}$$

Let $K_1 = K_0 \uparrow \eta_0$. Now, in order to formulate some objectives of the system, let a_j^o denote that agent a owns a resource of type j and similarly for the other agents. Let

$$\text{happy}(i) = \bigwedge_{R_j \in \text{Useful}(i)} i_j^o$$

Thus $\text{happy}(i)$ means that i is in possession of all his useful resources, simultaneously. Our first objective is:

$$\varphi_1 = A \square \bigwedge_{i \in A} A \diamond \text{happy}(i).$$

The normative system that we will use for it is

$$\eta_1 = \{(s, s') \mid \text{turn}(s) = i \ \& \ O_i = \text{Useful}(i) \ \& \ O'_i \neq \emptyset\}$$

In words: if at some point an agent simultaneously owns all the resources that are useful for him, then he will make them available if it is his turn. Which coalitions are sufficient for this norm in the context of K_1 and φ_1 ? First of all, consider a coalition without agent a . If a does not comply with norm η_1 , then he can grab the printer and hold on to it forever. Thus, agent b will not be happy, because there is only one printer. The same argument holds for a coalition without agent b . Thus, it seems that any sufficient coalition must include both agents a and b . But $\{a, b\}$ alone is not a sufficient coalition, as the following scenario illustrates: (1) a grabs a PC; (2) b grabs the printer; (3) c grabs a scanner; (4) d grabs the other scanner. Now, if c and d do not comply with η_1 , it might be that they never give up their scanners, in which case

