A Protocol for Resource Sharing in Norm-Governed Ad Hoc Networks

Alexander Artikis[1], Lloyd Kamara[2], Jeremy Pitt[2], and Marek Sergot[1]

[1] Department of Computing, SW7 2BZ
[2] Electrical & Electronic Engineering Department, SW7 2BT
Imperial College London
a.artikis@acm.org, {l.kamara, j.pitt, m.sergot}@imperial.ac.uk

Abstract. Ad hoc networks may be viewed as computational systems whose members may fail to, or choose not to, comply with the rules governing their behaviour. We are investigating to what extent ad hoc networks can usefully be described in terms of permissions, obligations and other more complex normative relations, based on our previous work on specifying and modelling open agent societies. We now propose to employ our existing framework for the management of ad hoc networks, exploiting the similarities between open agent societies and ad hoc networks viewed at the application level. We also discuss the prospects of modelling ad hoc networks at the physical level in similar terms. We demonstrate the framework by constructing an executable specification, in the event calculus, of a common type of protocol used to regulate the control of access to shared resources in ad hoc networks.

1 Introduction

Ad Hoc Network (*AHN*) is a term used to describe a transient association of network nodes which inter-operate largely independently of any fixed support infrastructure [17]. An AHN is typically based on wireless technology and may be short-lived, supporting spontaneous rather than long-term interoperation [18]. Such a network may be formed, for example, by the devices of the participants in a workshop or project meeting (for sharing and co-authoring documents); by consumers entering and leaving an 802.11 wireless hot spot covering a shopping mall (for buying/selling goods C2C-style by matching potential buyers and sellers); or by emergency or disaster relief workers, where the usual static support infrastructure is unavailable.

An AHN may be visualised as a continuously changing graph [17]: connection and disconnection may be controlled by the physical proximity of the nodes or, it may be controlled by the nodes' continued willingness to cooperate for the formation, and maintenance, of a cohesive (but potentially transient) community. An issue that typically needs to be addressed when managing and maintaining an AHN is that of resource sharing: the participating nodes compete over a set of limited resources such as bandwidth and power (for example, battery consumption). Often AHNs are specifically set up for sharing a resource such as broadband Internet access, processor cycles, file storage, or a document in the

project meeting example mentioned above. In all cases the limited resources are controlled by the participants of a network.

Due to its inherently transient nature, an AHN needs to be 'adaptable', that is, it should be able to deal with 'exceptions'. The aim of our present research is to investigate to what extent adaptability can be enhanced by viewing AHNs as instances of *norm-governed* systems. We want to examine this question both at the *application level* and at the *physical level*. At the application level, an AHN can be viewed as an *open agent society* [1–3], that is, a computational (agent) community exhibiting the following characteristics:

- Members are programmed by different parties — moreover, there is no direct access to a member's internal state and so we can only make inferences about that state.
- Members do not necessarily share a notion of global utility — they may fail to, or even choose not to, conform to the community specifications in order to achieve their individual goals.
- The members' behaviour and interactions cannot be predicted in advance.

In previous work [1–3] we presented a theoretical framework for specifying open agent societies in terms of concepts stemming from the study of legal and social systems. The behaviour of the members of an open agent society is regulated by rules expressing their *permissions*, *obligations* and other more complex normative relations that may exist between them [10]. Software tools enable formal specifications of these rules to be executed and analysed in various ways. We propose to use this framework for the management of AHNs. In this paper we focus on the issue of resource sharing and employ the theoretical framework to specify a common family of protocols for controlling access to shared resources.

We believe that there may also be value in viewing an AHN as an instance of a norm-governed system at the *physical level*. This is because it is possible, even likely, that system components will fail to behave as they ought to behave — not from wilfulness or to seek advantage over others but simply because of the inherently transient nature of the AHN. It is therefore meaningful to speak of system components failing to comply with their obligations, of permitted/forbidden actions, and even of 'sanctions' (though clearly not of 'punishments'). A secondary aim of our research is to investigate to what extent the methods we have previously used to model open agent societies can be applied to this new setting.

The remainder of this paper is divided into three main parts. First, we review a line of research on resource sharing, namely *floor control protocols*. Second, we present a specification of a protocol for resource sharing in norm-governed AHNs. The presentation of the protocol specification includes a description of the relevant parts of the theoretical framework mentioned above. Third, we summarise the presented work and outline directions for current and future research.

2 Floor Control Protocols

In the fields of Collaborative Multimedia Computing (CMC) and Computer-Supported Co-operative Work (CSCW), the term *floor control* denotes a service

guaranteeing that at any given moment only a designated set of users (subjects) may simultaneously work with or on the same objects (shared resources), thus creating a temporary exclusivity for access on such resources [5].

> "[...F]loor control lets users attain exclusive control over a shared resource by being granted the floor, extending the traditional notion as the "right to speak" [20] to the multimodality of data formats in networked multimedia systems. We understand floor control as a technology to implement group coordination, but use both terms synonymously in this paper." [7, p.18]

Sharing a resource may be achieved by executing *Floor Control Protocols* (*FCP*s) and *Session Control Protocols* (*SCP*s). FCPs prescribe ways for mutually exclusive access to shared resources amongst the subjects. A number of properties of such protocols have been identified [5,6]: *safety* (each floor request is eventually serviced), *fairness* (no subject 'starves', each floor request is serviced based on a common metric), and so on. SCPs prescribe ways for, amongst other things, joining a FCP (or session), withdrawing from a session, inviting to join a session, determining the resources to be shared, and determining the *policy* of a session, that is, the ways in which a floor may be requested or granted. Example policies are *chair-designated* (an elected participant is the arbiter over the usage of specific floors), *election* (participants vote on the next subject holding the floor), and *lottery scheduling* (floor assignment operates on a probabilistic basis).

It is our assumption that the abstractions of floor control and session control are applicable to the issue of resource sharing in AHNs. Clarifying what 'being granted the floor' or 'holding the floor' implies is one of the aims of the formalisation presented in later sections. The concept of session control (or *conference management* [23]) is applicable to the formation of an AHN, and to the management and maintenance of such a network in general. In this paper, however, we will focus on the issue of floor control, assuming that an AHN and a FCP within that network have already been established. The issue of session control will be addressed elsewhere.

We will present a specification of a simple *chaired Floor Control Protocol* (*cFCP*). (We apologise for the unfortunate mixed metaphor.) The chair-designated policy was chosen simply to provide a concrete example of a FCP — we could have equally chosen an election, or some other policy type. Moreover, we have intentionally omitted to address several of the design issues set out in the literature on FCPs (for instance, that a protocol should provide mutually exclusive resource access in 'real-time' [5,6]). Our point here is to illustrate that, in settings in which subjects (or other system members) may fail to behave as they ought to behave, any protocol specification for resource sharing (following a chair-designated, lottery scheduling or any other policy type, stemming from the CSCW, CMC, or any other research field) needs to express what a member is permitted to do, obliged to do, and, possibly, additional normative relations that may exist between them.

Two factors that characterise a FCP are [7]: (i) the mechanism and node topology that determine the ways in which floor information (for instance, floor

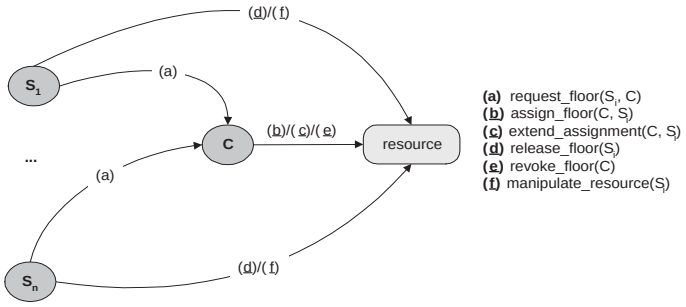

Fig. 1. A two-role chaired floor control protocol.

requests, the status of the floor, and so on) is communicated amongst the participants, and (ii) the policy followed in the protocol. Factor (i) is the major design decision for a group coordination protocol and determines, amongst other things, which policies are established in a protocol. We adopt a high-level view of FCPs: we specify the rules prescribing the ways in which a floor is requested and granted without making any explicit assumptions about the node topology and distribution of floor information in general.

3 A Protocol for Resource Sharing in Ad Hoc Networks

In this section we present a chaired Floor Control Protocol (cFCP). For simplicity, we present a cFCP specification concerning a single resource, a single floor (associated with the resource), and a single chair, that is, a distinguished participant determining which other participant is actually given the floor. In this setting, the allocation of several resources in an AHN may be performed by several parallel executions of FCPs (following a chair-designated, election or any other policy type). Our cFCP specification includes the following roles:

- *Subject*, the role of designated participants performing the following actions: *request_floor* (requesting the floor from the chair), *release_floor* (releasing the floor), and *manipulate_resource* (physically manipulating the resource). Sometimes we will refer to the subject holding the floor as a 'holder'.
- *Chair*, the controller for the floor, that is, the participant performing the following actions: *assign_floor* (assigning the floor for a particular time period to a subject), *extend_assignment* (extending the time for which the floor may be held), and *revoke_floor* (revoking the floor from the holder).

The floor can be in one of the following states: (i) *granted*, denoting that a subject has been given exclusive access to the resource by the chair, or (ii) *free*, denoting that no subject currently holds the floor. In both cases, the floor may or may not be requested by a subject (for example, the floor may be granted to subject S' and requested by subject S'' at the same time). We make the following comments concerning our cFCP specification. First, there are no time-outs

(deadlines) prescribing when a request should be issued, a floor should be assigned, or an assignment should be extended. Second, there is no termination condition signalling the end of the protocol. There is no particular difficulty in including timeouts and termination conditions in the formalisation but it lengthens the presentation and is omitted here for simplicity. See [2, 3] for example formalisations of deadlines and termination conditions in the context of protocol specifications.

Figure 1 provides an informal presentation of the possible interactions between the entities of a cFCP, that is, the subjects $S_1, \ldots, S_n$, the chair C, and the resource. The actions of our protocol specification may be classified into two categories: (i) communicative actions and, (ii) physical actions. The first category includes the *request_floor* action whereas the second category includes the *assign_floor*, *extend_assignment*, *release_floor*, *revoke_floor*, and *manipulate_resource* actions[3]). Consider an example in which the shared resource is hard disk space. In this setting, the action of assigning the floor could be realised as creating an account on the file server so that the holder can manipulate the resource, that is, store files.

4 An Event Calculus Specification

In previous work we employed three action languages with direct routes to implementation to express protocol specifications:

1. The Event Calculus (EC) [13], a formal, intuitive and well-studied action language (see [2] for an EC specification of a contract-net protocol).
2. The $C+$ language [9], a formalism with explicit transition systems semantics (see [3] for a $C+$ specification of a dispute resolution protocol).
3. The $(C+)^{++}$ language [25], an extended form of $C+$ specifically designed for modelling the normative and institutional aspects of multi-agent systems (see [25] for a $(C+)^{++}$ specification of a resource sharing protocol).

Each formalism has its advantages and disadvantages (see [1, Section 6.12] for a discussion about the utility of $C+$, $(C+)^{++}$ and EC on protocol specification). In this paper we will use EC because an EC implementation (in terms of logic programming) has proved to be more efficient than a $C+$ or $(C+)^{++}$ implementation (in terms of the *Causal Calculator*, a software tool supporting computational tasks regarding the $C+$ language) for the provision of 'run-time services' (a description of such services is presented in Section 6).

First, we briefly present EC. Second, we specify the *social constraints* (or protocol rules) governing the behaviour of the cFCP participants. We maintain the standard and long established distinction between *physical capability*, *institutionalised power* and *permission* (see, for instance, [11, 15] for illustrations of this distinction). Accordingly, our specification of social constraints expresses: (i) the externally observable physical capabilities, (ii) institutional powers, and (iii)

[3] The following convention is adopted in the figures of this paper: physical actions are represented by an underlined letter (for example, (b)) whereas communicative actions are represented with no underlining (for example, (a)).

Table 1. Main Predicates of the Event Calculus.

Predicate	Meaning
happens(Act, T)	Action Act occurs at time T
initially($F = V$)	The value of fluent F is V at time 0
holdsAt($F = V$, T)	The value of fluent F is V at time T
initiates(Act, $F = V$, T)	The occurrence of action Act at time T initiates a period of time for which the value of fluent F is V
terminates(Act, $F = V$, T)	The occurrence of action Act at time T terminates a period of time for which the value of fluent F is V

permissions and obligations of the cFCP participants; in addition, it expresses (iv) the *sanctions* and *enforcement policies* that deal with the performance of forbidden actions and non-compliance with obligations.

4.1 The Event Calculus

The Event Calculus (EC), introduced by Kowalski and Sergot [13], is a formalism for representing and reasoning about actions or events and their effects in a logic programming framework. In this section we briefly describe the version of the EC that we employ. EC is based on a many-sorted first-order predicate calculus. For the version used here, the underlying time model is linear and it may include real numbers or integers. Where F is a *fluent* (a property that is allowed to have different values at different points in time), the term $F = V$ denotes that fluent F has value V. Boolean fluents are a special case in which the possible values are true and false. Informally, $F = V$ holds at a particular time-point if $F = V$ has been *initiated* by an action at some earlier time-point, and not *terminated* by another action in the meantime.

An *action description* in EC includes axioms that define, amongst other things, the action occurrences (with the use of happens predicates), the effects of actions (with the use of initiates and terminates predicates), and the values of the fluents (with the use of initially and holdsAt predicates). Table 1 summarises the main EC predicates. Variables (starting with an upper-case letter) are assumed to be universally quantified unless otherwise indicated. Predicates, function symbols and constants start with a lower-case letter. The domain-independent definitions of the EC predicates are presented in the Appendix. In the following sections we present an EC action description expressing our cFCP specification.

4.2 Physical Capability

Table 2 displays a number of the fluents of the EC action description expressing the cFCP specification. The utility of these fluents will be explained during the

Table 2. Main Fluents of the cFCP Specification.

Fluent	Domain	Textual Description
$requested(S, T)$	boolean	subject S requested the floor at time T
$status$	$\{free, granted(S, T)\}$	the status of the floor: $status = free$ denotes that the floor is free whereas $status = granted(S, T)$ denotes that the floor is granted to subject S until time T
$best\_candidate$	agent identifiers	the best candidate for the floor
$can(Ag, Act)$	boolean	agent Ag is capable of performing Act
$pow(Ag, Act)$	boolean	agent Ag is empowered to perform Act
$per(Ag, Act)$	boolean	agent Ag is permitted to perform Act
$obl(Ag, Act)$	boolean	agent Ag is obliged to perform Act
$sanction(Ag)$	$\mathbb{Z}^*$	the sanctions of agent Ag

Table 3. Physical Capability and Institutional Power in the cFCP.

Action	can	pow
$assign\_floor(C, S)$	$status = free$	–
$extend\_assignment(C, S)$	$status = granted(S, T)$	–
$revoke\_floor(C)$	$status = granted(S, T)$	–
$release\_floor(S)$	$status = granted(S, T)$	–
$manipulate\_resource(S)$	$status = granted(S, T)$	–
$request\_floor(S, C)$	$\top$	not $requested(S, T)$

presentation of the protocol specification. This section presents the specification of the externally observable physical capabilities of the cFCP participants. The second column of Table 3 presents the conditions that, when satisfied, enable the participants to perform the actions displayed in the first column of this table. We will refer to these conditions as expressing 'physical capability' though the term 'practical possibility' might have been employed instead. (In Table 3, C represents an agent occupying the role of the chair and S represents an agent occupying the role of the subject.)

The chair is capable of assigning the floor to a subject if and only if the floor is free (see Table 3). The performance of such an action always changes the status of the floor as follows:

$$\begin{aligned} &\mathsf{initiates}(assign\_floor(C, S), status = granted(S, T'), T) \leftarrow \\ &\qquad role\_of(C, chair),\ role\_of(S, subject), \\ &\qquad \mathsf{holdsAt}(status = free, T),\ (T' := T + 5) \end{aligned} \tag{1}$$

After assigning the floor to a subject S at time T, the floor is considered granted until some future time (say $T+5$). The first two conditions of axiom (1) refer to the roles of the participants. We assume (in this version) that the participants of a cFCP do not change roles during the execution of a protocol, and so *role_of* is treated as an ordinary predicate and not as a time-varying fluent. Notice that the practical capability condition is included here as part of the initiates specification. There are other possible treatments of the practical capability conditions but we do not have space for discussion of alternative treatments here.

Note also that the chair can assign the floor to a subject that has never requested it. In some systems, this type of behaviour may be considered 'undesirable' or 'wrong'. Section 4.4 presents how 'undesirable' behaviour in the cFCP is specified by means of the concept of *permitted* action.

If an assignment concerns a subject that has requested the floor, represented by the *requested* fluent (see Table 2), then this request is considered serviced, that is, the associated *requested* fluent no longer holds:

$$\begin{aligned} &\mathsf{initiates}(assign\_floor(C,S), requested(S,T') = \mathsf{false}, T) \leftarrow \\ &\quad role\_of(C, chair), \\ &\quad \mathsf{holdsAt}(status = free, T), \\ &\quad \mathsf{holdsAt}(requested(S,T') = \mathsf{true}, T) \end{aligned} \tag{2}$$

The chair can extend the assignment of the floor to a subject S if and only if S is holding the floor. Moreover, extending the assignment of the floor changes its status as follows:

$$\begin{aligned} &\mathsf{initiates}(extend\_assignment(C,S), status = granted(S,T''), T) \leftarrow \\ &\quad role\_of(C, chair), \\ &\quad \mathsf{holdsAt}(status = granted(S,T'), T),\ (T'' := T' + 5) \end{aligned} \tag{3}$$

In other words, if the floor was granted to S until time T', after the extension it will be granted until time $T' + 5$. Note that the chair is capable of extending the floor even if the holder has not requested such an extension.

A subject S can release the floor if and only if S is the holder (irrespective of whether or not the allocated time for the floor has ended). Releasing the floor changes its status as follows:

$$\begin{aligned} &\mathsf{initiates}(release\_floor(S), status = free, T) \leftarrow \\ &\quad \mathsf{holdsAt}(status = granted(S,T'), T) \end{aligned} \tag{4}$$

In a similar manner we express when an agent is capable of performing the remaining physical actions of the protocol as well as the effects of these actions.

In this example cFCP there is only one communicative action, that of requesting the floor. We have specified that a subject is always physically capable of communicating a request for the floor to the chair. For the specification of the effects of this action, it is important to distinguish between the act of ('successfully') issuing a request and the act by means of which that request is issued.

Communicating a request for the floor, by means of sending a message of a particular form via a TCP/IP socket connection, for example, is not necessarily 'successful', in the sense that the request is eligible to honoured by the chair. It is only if the request is communicated by an agent with the *institutional power* to make the request that it will be 'successful'. An account of institutional power is presented in the following section.

4.3 Institutional Power

The term institutional (or 'institutionalised') power refers to the characteristic feature of organisations/institutions — legal, formal, or informal — whereby designated agents, often when acting in specific roles, are empowered, by the institution, to create or modify facts of special significance in that institution — *institutional facts* — usually by performing a specified kind of act. Searle [24], for example, has distinguished between *brute facts* and institutional facts. Being in physical possession of an object is an example of a brute fact (it can be observed); being the owner of that object is an institutional fact.

According to the account given by Jones and Sergot [11], institutional power can be seen as a special case of a more general phenomenon whereby an action, or a state of affairs, A — because of the rules and conventions of an institution — counts, in that institution, as an action or state of affairs B (such as when sending a letter with a particular form of words counts as making an offer, or banging the table with a wooden mallet counts as declaring a meeting closed).

In some circumstances it is unnecessary to isolate and name all instances of the acts by means of which agents exercise their institutional powers. It is convenient to say, for example, that 'the subject S requested the floor from the chair C' and let the context disambiguate whether we mean by this that S performed an action, such as sending a message of a particular form via a TCP/IP socket connection, by means of which the request for the floor is signalled, or whether S actually issued a request, in the sense that this request is eligible to be honoured by C. We disambiguate in these circumstances by attaching the label 'valid' to act descriptions. We say that an action is *valid* at a point in time if and only if the agent that performed that action had the *institutional power* (or just 'power' or 'was empowered') to perform it at that point in time. So, when we say that 'the subject S requested the floor from the chair C' we mean, by convention, merely that S signalled its intention to request the floor; this act did not necessarily constitute the request eligible to be honoured. In order to say that a request is eligible to be honoured, we say that the action 'subject S requested the floor' was *valid*: not only did S signal its intention to request the floor, but also S had the institutional power to make the request. Similarly, *invalid* is used to indicate lack of institutional power: when we say that the action 'subject S requested the floor' is invalid we mean that S signalled its intention to request it but did not have the institutional power to do so at that time (and so the attempt to make the request eligible to be serviced was not successful).

We express the institutional power to request the floor as follows:

$$\begin{aligned} \mathsf{holdsAt}(pow(S, request\_floor(S, C)) = \mathsf{true}, T) \leftarrow \\ role\_of(C, chair),\ role\_of(S, subject), \\ \mathsf{not}\ \mathsf{holdsAt}(requested(S, T') = \mathsf{true}, T) \end{aligned} \tag{5}$$

Axiom (5) expresses that a subject S is empowered to request the floor from the chair C if S has no pending valid requests. not represents *negation by failure* [4].

The existence of a valid request is recorded with the use of the *requested* fluent:

$$\begin{aligned} \mathsf{initiates}(request\_floor(S, C), requested(S, T) = \mathsf{true}, T) \leftarrow \\ \mathsf{holdsAt}(pow(S, request\_floor(S, C)) = \mathsf{true}, T) \end{aligned} \tag{6}$$

There is no corresponding fluent for invalid requests.

4.4 Permission and Obligation

Now we specify which of the cFCP acts are permitted or obligatory. Behaviour which does not comply with the specification is regarded as 'undesirable'. Such behaviour is not necessarily wilful. When an AHN member performs a non-permitted act or fails to perform an obligatory act, it could be deliberate, as when an agent (at the application level) seeks to gain an unfair advantage, but it could also be unintentional, and it could even be unavoidable, due to network conditions outside that member's control.

The definitions of permitted actions are application-specific. It is worth noting that there is no fixed relationship between powers and permissions. In some computational societies an agent is permitted to perform an action if that agent is empowered to perform that action. In general, however, an agent can be empowered to perform an action without being permitted to perform it (perhaps temporarily). The specification of obligations is also application-specific. It is important, however, to maintain the consistency of the specification of permissions and obligations: an agent should not be forbidden and obliged to perform the same action at the same time.

Table 4 displays the conditions that, when satisfied, oblige or simply permit the cFCP participants to perform an action. (In this table, *CurrentTime* represents the time that the presented conditions are evaluated.) There are other possible specifications of permitted and obligatory actions. The presented ones were chosen simply to provide a concrete illustration of cFCP.

The chair is permitted and obliged to assign the floor to a subject S provided that: (i) the floor is free, and (ii) S is the best candidate for the floor (see Table 4). The procedure calculating the best candidate for the floor at each point in time is application-specific. For the sake of this example, the best candidate is defined to be the one with the earliest (valid) request. In more realistic scenarios the calculation of the best candidate would consider additional factors such as how urgent the request is, how many times the requesting subject had the floor

Table 4. Permission and Obligation in the cFCP.

Action	per	obl
$assign\_floor(C, S)$	$status = free$, $best\_candidate = S$	$status = free$, $best\_candidate = S$
$extend\_assignment(C, S)$	$status = granted(S, T)$, $best\_candidate = S$	$status = granted(S, T)$, $best\_candidate = S$
$revoke\_floor(C)$	$status = granted(S, T)$, $CurrentTime \geq T$, $best\_candidate \neq S$	$status = granted(S, T)$, $CurrentTime \geq T$, $best\_candidate = S'$, $S \neq S'$
$release\_floor(S)$	$\top$	$status = granted(S, T)$, $CurrentTime \geq T$, $best\_candidate = S'$, $S \neq S'$
$manipulate\_resource(S)$	$status = granted(S, T)$, $CurrentTime < T$	$\bot$
$request\_floor(S, C)$	$\top$	$\bot$

in the past, and so on[4]. There is no difficulty in expressing such definitions in the formalism employed here. Indeed, the availability of the full power of logic programming is one of the main attractions of employing EC as the temporal formalism.

According to the above specification of permission, when the floor is free the chair is only permitted to assign it to the best candidate (if any). At the same time, however, the chair is capable of assigning it to any subject participating in the cFCP (see Table 3).

The chair is permitted to revoke the floor if: (i) the floor is currently granted to a subject, (ii) the allocated time for the floor has ended, and (iii) the subject holding the floor is currently not the best candidate for the floor. Note that the chair can revoke the floor even if the allocated time for the floor has not ended or if the subject holding the floor is currently the best candidate for it.

The chair is permitted to revoke the floor (after the allocated time for the holder has ended) even if there is no subject requesting the floor. If there is a subject requesting the floor, however, and that subject is the best candidate, then the chair is not only permitted, but obliged to revoke the floor:

$$\begin{aligned} &\mathsf{holdsAt}(obl(C, revoke\_floor(C)) = \mathsf{true}, T) \leftarrow \\ &\quad role\_of(C, chair), \\ &\quad \mathsf{holdsAt}(status = granted(S, T'), T),\ (T \geq T'), \\ &\quad \mathsf{holdsAt}(best\_candidate = S', T),\ (S \neq S') \end{aligned} \tag{7}$$

[4] The best candidate is picked from the set of subjects having pending (valid) requests, not from the set of all subjects participating in a cFCP.

We have chosen to specify that a subject is always permitted to release the floor, although releasing the floor is not always physically possible. Alternatively, we could have specified that the permission to release the floor coincides with the physical capability to do so. In this example, we might guess that the alternatives are equivalent, in the sense that they produce protocols that always have the same outcome. This is a hypothesis that can be tested. One aim of the work presented here is to provide computational tools to support the automated testing of such hypotheses.

A subject S is permitted to manipulate the resource if S is holding the floor and the allocated time for it has not ended. Permitted or not, S is never obliged to manipulate the resource. Similarly, a subject is never obliged to request the floor — it is always permitted, however, to do so.

4.5 Sanction

Sanctions and enforcement policies are a means of dealing with 'undesirable' behaviour. In the cFCP, we want to reduce or eliminate the following types of 'undesirable' behaviour:

- the chair extending the assignment of, and revoking the floor when being forbidden to do so, and
- non-compliance with the obligation to assign, revoke and release the floor.

One possible enforcement strategy is to try to devise additional controls (physical or institutional) that will force agents to comply with their obligations or prevent them from performing forbidden actions. When competing for hard disk space, for example, a forbidden revocation of the floor may be physically blocked, in the sense that it is not possible to delete the holder's account on the file server. The general strategy of designing mechanisms to force compliance and eliminate non-permitted behaviour is what Jones and Sergot [10] referred to as *regimentation*. Regimentation devices have often been employed in order to eliminate 'undesirable' behaviour in computational systems. *Interagents* [21], for example, enforce the rules of the FishMarket auction house to the buyer and seller agents. *Sentinels* [12] monitor and, when necessary, modify some aspects of the agent interactions in order to provide 'exception handling' services. *Controllers* [16] enforce the 'law-governed interaction' coordination mechanism in open agent societies. (Lomuscio and Sergot [14] show how it is possible to determine formally whether the introduction of a controller does have the intended effect of eliminating unwanted system behaviour.) It has been argued [10], however, that regimentation is rarely desirable (it results in a rigid system that may discourage agents from entering it [19]), and not always practical. The practicality of regimentation devices is even more questionable when considering AHNs, due to the transient nature of these networks. In any case, violations may still occur even when regimenting a computational system (consider, for instance, a faulty regimentation device). For all of these reasons, we have to allow for sanctioning and not rely exclusively on regimentation mechanisms.

For the present example, we employ an additive fluent, $sanction(Ag)$, to express each participant's sanctions (see Table 2): initially, the value of this

fluent is equal to zero and it is incremented every time a participant exhibits the type of 'undesirable' behaviour mentioned above. Consider the following example: the chair is sanctioned if it assigns the floor to a subject S while it is obliged to assign the floor to another subject S':

$$\begin{aligned}\mathsf{initiates}(&assign\_floor(C,S), sanction(C)=V',T) \leftarrow \\ &role\_of(S, subject), \\ &\mathsf{holdsAt}(obl(C, assign\_floor(C,S'))=\mathsf{true},T),\ (S \neq S'), \\ &\mathsf{holdsAt}(sanction(C)=V,T),\ (V' := V+1)\end{aligned} \tag{8}$$

According to axiom (8), every time the chair C fails to comply with its obligation to assign the floor the value of the associated $sanction(C)$ fluent is incremented by one. Similarly, we update the value of $sanction(Ag)$ when the remaining participants exhibit 'undesirable' behaviour. We would ordinarily also include a means for decreasing the value of a $sanction(Ag)$ fluent, for instance if Ag has not performed forbidden ('undesirable') actions for a specified period of time. We have omitted the details for simplicity of the presentation.

One way of discouraging the performance of forbidden actions and non-compliance with obligations (at the application level) is by penalising this type of behaviour. We specify the following penalties for the aforementioned sanctions (the presented specification is but one of the possible approaches, chosen here merely to provide a concrete illustration). Consider the following example:

$$\begin{aligned}\mathsf{holdsAt}(&pow(S, request\_floor(S,C))=\mathsf{true},T) \leftarrow \\ &role\_of(C, chair),\ role\_of(S, subject), \\ &\mathsf{not\ holdsAt}(requested(S,T')=\mathsf{true},T), \\ &\mathsf{holdsAt}(sanction(S)=V,T),\ (V<5)\end{aligned} \tag{5'}$$

The above formalisation is a modification of the axiom expressing the power to request the floor (that is, axiom (5)), in the sense that it considers the sanctions associated with a subject S: when the value of $sanction(S)$ is greater or equal to five (say) then S is no longer empowered to request the floor. One may argue that once that happens, S is no longer an 'effective' participant of the protocol, in the sense that S may no longer 'successfully' request the floor. It may be the case, however, that the chair does not abide by the protocol rules and assigns (and even extends the assignment of) the floor to S, even though S has not 'successfully' requested the floor.

We anticipate applications in which agents participate in a Session Control Protocol (SCP) before taking part in a cFCP in order to acquire a set of roles that they will occupy while being part of that cFCP. Given the value of $sanction(C)$, a chair C may be:

- *suspended*, that is, C is temporarily disqualified from acting as a chair in future cFCPs. More precisely, C may not 'effectively' participate, for a specified period, in a SCP and, therefore, may not acquire the role of the chair.
- *banned*, that is, C is permanently disqualified from acting as a chair.

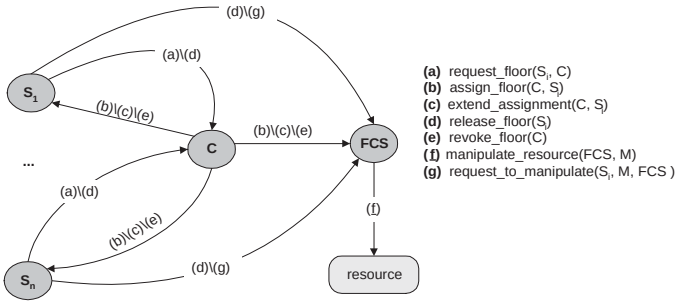

Fig. 2. A three-role chaired floor control protocol.

Being deprived of the role of the chair means, in this example, being deprived of the permission and, more importantly, the physical capability to assign, extend the assignment of, and revoke the floor. Alternatively, a sanctioned chair may be suspended or banned from acting as a subject in future FCPs (not necessarily chaired-designated ones), thus not being able to compete for, and access other shared resources in an AHN. The axiomatisation of the penalties associated with a sanctioned chair and a detailed discussion about SCPs in general will be presented elsewhere (see, however, [2, Section 3.2], [1, Section 4.5]) for a brief presentation of role-assignment in open agent societies).

At the physical level, where the members of the AHN are network devices, the question of imposing penalties clearly does not arise. There is a possible role for 'sanctions' nevertheless. In the present example, the value of the $sanction(Ag)$ fluent can be seen as a measure of Ag's 'reliability'. When the value of that fluent passes the specified threshold, floor assigning capabilities (say) may be suspended (and usually passed to another network member) not as a 'punishment' but as a way of adapting the network organisation. To what extent this view gives useful insights in practice is a topic of our current research.

5 A Few Notes on cFCP

In the FCP literature, a cFCP usually includes a third role, that of the *Floor Control Server* (*FCS*) [23]. Figure 2 provides an informal presentation of the possible interactions between the entities of a three-role cFCP. In such a setting, only the FCS can physically manipulate the resource. A subject holding the floor may only *request* from the FCS to manipulate the resource, describing the type of manipulation M — it is up to the FCS whether this request will be honoured or not. The chair still assigns, extends the assignment of, and revokes the floor. These actions, however, are now communicative ones, they are multi-casted to the holder and the FCS. Similarly, releasing the floor is now a communicative action, multi-casted to the chair and the FCS.

In order to illustrate the difference between the two-role and three-role cFCP, we outline the physical capabilities and institutional powers associated with a holder in each setting. In a two-role cFCP, a holder S has the physical capability to manipulate the shared resource. In a three-role cFCP, a holder S has the institutional power to request (from the FCS) to manipulate the shared resource. Unlike the two-role setting, in a three-role cFCP a holder may not succeed in manipulating the shared resource (for example, if the FCS disregards S's valid requests for manipulation of the resource, thus not complying with the protocol rules). Developing a complete specification of a three-role cFCP and comparing that with a specification of a two-role cFCP is another topic of our current research.

6 Discussion

We have presented a specification of a simple protocol for resource sharing in norm-governed AHNs. The specification of norm-governed computational systems has been the focus of several studies stemming from various research fields. A few examples are [8, 16, 21, 26–28]. Generally, work on the specification of norm-governed computational systems does not explicitly represent the institutional powers of the member agents. This is one key difference between our work and related approaches in the literature: our specification of social constraints explicitly represents the institutional powers of the agents, differentiates between institutional power, permission, physical capability and sanction, and employs formalisms with a declarative semantics and clear routes to implementation to express these concepts. (A detailed comparison between our work and related approaches in the literature can be found in [1, Section 4.10].)

The cFCP specification is expressed as a logic program and is therefore directly executable providing a clear route to (prototype) implementations. In previous work [2] we presented ways of executing an EC action description expressing a protocol specification. The cFCP executable specification may inform the agents' decision-making at run-time, for example, by allowing the powers, permissions, obligations, and sanctions current at any time to be determined.

At design-time, agents may wish to prove various properties of the protocol specification in order to decide whether or not they should participate in the protocol. Such properties may include, for instance, that a protocol specification is 'safe' and 'fair' (see Section 2), that no agent is forbidden and obliged to perform an action at the same time, non-compliance with the obligation to assign the floor always leads to a sanction, and so on. We have been experimenting [3,25] with the use of various techniques (for example, planning query computation and model checking) to prove properties of a protocol specification expressed in the $C+$ language. (Our theoretical framework for specifying norm-governed systems is not dependent on any particular action language or temporal structure.) We aim to investigate the feasibility and practicality of the application of some of the aforementioned techniques to an EC-formalised protocol specification in order to prove properties of such a specification.

Sadighi and Sergot [22] argue that when dealing with resource access control in heterogeneous computational systems in which 'undesirable' behaviour may arise (such as AHNs), the concepts of permission and prohibition are inadequate and need to be extended with that of *entitlement*: "entitlement to access a resource means not only that the access is permitted but also that the controller of the resource is obliged to grant the access when it is requested" [22]. We are currently working towards a treatment of this and related senses of 'entitlement' as they arise in the context of our cFCP specification (entitlement is concept that arises naturally in a three-role cFCP). More precisely, we are identifying the conditions in which a subject holding the floor can be said to be 'entitled' to it, and what the consequences are, and the circumstances in which it is meaningful to say that a subject not holding the floor is 'entitled'/not 'entitled' to it, and what the consequences are.

We believe that viewing AHNs as instances of norm-governed systems enhances their 'adaptability' both at the application level and at the physical level. By specifying the permissions, obligations, entitlements, and other more complex normative relations that may exist between the members of an AHN, one may precisely identify 'undesirable' behaviour, such as performance of forbidden actions and non-compliance with obligations, and, therefore, introduce enforcement strategies in order to adapt to such behaviour. To what extent this view gives useful insights in practice remains to be investigated.

Acknowledgements

This work has been supported by the EPSRC project "Theory and Technology of Norm-Governed Self-Organising Networks" (GR/S74911/01).

References

1. A. Artikis. *Executable Specification of Open Norm-Governed Computational Systems*. PhD thesis, University of London, November 2003. Retrieved April 8, 2004, from `http://www.doc.ic.ac.uk/~aartikis/publications/artikis-phd.pdf`, also available from the author.
2. A. Artikis, J. Pitt, and M. Sergot. Animated specifications of computational societies. In *Proceedings of Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1053–1062. ACM Press, 2002.
3. A. Artikis, M. Sergot, and J. Pitt. An executable specification of an argumentation protocol. In *Proceedings of International Conference on Artificial Intelligence and Law (ICAIL)*, pages 1–11. ACM Press, 2003.
4. K. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 293–322. Plenum Press, 1978.
5. H.-P. Dommel and J. J. Garcia-Luna-Aceves. Design issues for floor control protocols. In *Proceedings of Symposium on Electronic Imaging: Multimedia and Networking*, volume 2417, pages 305–316. IS&T/SPIE, 1995.
6. H.-P. Dommel and J. J. Garcia-Luna-Aceves. Floor control for multimedia conferencing and collaboration. *Multimedia Systems*, 5(1):23–38, 1997.

7. H.-P. Dommel and J. J. Garcia-Luna-Aceves. Efficacy of floor control protocols in distributed multimedia collaboration. *Cluster Computing Journal, Special issue on Multimedia Collaborative Environments*, 2(1):17–33, 1999.
8. M. Esteva, J. Rodriguez-Aguilar, C. Sierra, P. Garcia, and J. Arcos. On the formal specifications of electronic institutions. In F. Dignum and C. Sierra, editors, *Agent Mediated Electronic Commerce*, LNAI 1991, pages 126–147. Springer, 2001.
9. E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1–2):49–104, 2004.
10. A. Jones and M. Sergot. On the characterisation of law and computer systems: the normative systems perspective. In *Deontic Logic in Computer Science: Normative System Specification*, pages 275–307. J. Wiley and Sons, 1993.
11. A. Jones and M. Sergot. A formal characterisation of institutionalised power. *Journal of the IGPL*, 4(3):429–445, 1996.
12. M. Klein, J. Rodriguez-Aguilar, and C. Dellarocas. Using domain-independent exception handling services to enable robust open multi-agent systems: the case of agent death. *Journal of Autonomous Agents and Munti-Agent Systems*, 7(1–2):179–189, 2003.
13. R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–96, 1986.
14. A. Lomuscio and M. Sergot. A formulation of violation, error recovery, and enforcement in the bit transmission problem. *Journal of Applied Logic*, 2:93–116, 2004.
15. D. Makinson. On the formal representation of rights relations. *Journal of Philosophical Logic*, 15:403–425, 1986.
16. N. Minsky and V. Ungureanu. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 9(3):273–305, 2000.
17. A. Murphy, G.-C. Roman, and G. Varghese. An exercise in formal reasoning about mobile communications. In *Proceedings of Workshop on Software Specification and Design*, pages 25–33. IEEE Computer Society, 1998.
18. C. Perkins. *Ad Hoc Networking*, chapter 1. Addison Wesley Professional, 2001.
19. H. Prakken. Formalising Robert's rules of order. Technical Report 12, GMD – German National Research Center for Information Technology, 1998.
20. H. Robert. *Robert's Rules of Order: The Standard Guide to Parliamentary Procedure*. Bantam Books, 1986.
21. J. Rodriguez-Aguilar, F. Martin, P. Noriega, P. Garcia, and C. Sierra. Towards a test-bed for trading agents in electronic auction markets. *AI Communications*, 11(1):5–19, 1998.
22. B. Sadighi and M. Sergot. Contractual access control. In *Proceedings of Workshop on Security Protocols*, 2002.
23. H. Schulzrinne. Requirements for floor control protocol. Internet Engineering Task Force, January 2004. Retrieved April 8, 2004, from `http://www.ietf.org/internet-drafts/draft-ietf-xcon-floor-control-req-00.txt`.
24. J. Searle. What is a speech act? In A. Martinich, editor, *Philosophy of Language*, pages 130–140. Oxford University Press, third edition, 1996.
25. M. Sergot. Modelling unreliable and untrustworthy agent behaviour. In *Proceedings of Workshop on Monitoring, Security, and Rescue Techniques in Multiagent Systems (MSRAS)*, Advances in Soft Computing. Springer-Verlag, 2004.
26. M. Singh. A social semantics for agent communication languages. In F. Dignum and M. Greaves, editors, *Issues in Agent Communication*, LNCS 1916, pages 31–45. Springer, 2000.

27. W. Vasconcelos. Norm verification and analysis of electronic institutions. In J. Leite, A. Omicini, P. Torroni, and P. Yolum, editors, *This Volume*. 2005.
28. M. Winikoff, W. Liu, and J. Harland. Enhancing commitment machines. In J. Leite, A. Omicini, P. Torroni, and P. Yolum, editors, *This Volume*. 2005.

Appendix: The Event Calculus

The domain-independent definition of the holdsAt predicate is as follows:

$$\begin{aligned} &\mathsf{holdsAt}(F=V,T) \leftarrow \\ &\quad \mathsf{initially}(F=V), \\ &\quad \mathsf{not\ broken}(F=V,0,T) \end{aligned} \tag{9}$$

$$\begin{aligned} &\mathsf{holdsAt}(F=V,T) \leftarrow \\ &\quad \mathsf{happens}(Act,T'), \\ &\quad T'<T, \\ &\quad \mathsf{initiates}(Act,F=V,T'), \\ &\quad \mathsf{not\ broken}(F=V,T',T) \end{aligned} \tag{10}$$

According to axiom (9) a fluent holds at time T if it held initially (time 0) and has not been 'broken' in the meantime, that is, terminated between times 0 and T. Axiom (10) specifies that a fluent holds at a time T if it was initiated at some earlier time T' and has not been terminated between T' and T. not represents negation by failure. The domain-independent predicate broken is defined as follows:

$$\begin{aligned} &\mathsf{broken}(F=V,T_1,T_3) \leftarrow \\ &\quad \mathsf{happens}(Act,T_2), \\ &\quad T_1 \leq T_2,\ T_2<T_3, \\ &\quad \mathsf{terminates}(Act,F=V,T_2) \end{aligned} \tag{11}$$

$F=V$ is 'broken' between T_1 and T_3 if an event takes place in that interval that terminates $F=V$. A fluent cannot have more than one value at any time. The following domain-independent axiom captures this feature:

$$\begin{aligned} &\mathsf{terminates}(Act,F=V,T) \leftarrow \\ &\quad \mathsf{initiates}(Act,F=V',T), \\ &\quad V \neq V' \end{aligned} \tag{12}$$

Axiom (12) states that if an action Act initiates $F=V'$ then Act also terminates $F=V$, for all other possible values V of the fluent F. We do not insist that a fluent must have a value at every time-point. In this version of EC, therefore, there is a difference between initiating a Boolean fluent $F=\mathsf{false}$ and terminating $F=\mathsf{true}$: the first implies, but is not implied by, the second.

We make two further comments regarding this version of EC. First, the domain-independent EC axioms, that is, axioms (9)–(12), specify that a fluent does not hold at the time that was initiated but holds at the time it was terminated. Second, in addition to their domain-independent definitions, the holdsAt and terminates predicates may be defined in a domain-dependent manner (see, for example, axioms (5) and (7)). The happens, initially and initiates predicates are defined only in a domain-dependent manner.