Dynamic Specifications for Norm-Governed Systems

Alexander Artikis[12], Dimosthenis Kaponis[2], and Jeremy Pitt[2]

[1] Institute of Informatics and Telecommunications,
National Centre for Scientific Research "Demokritos", Athens 15310
[2] Electrical & Electronic Engineering Department,
Imperial College London, SW7 2BT
a.artikis@acm.org, dkaponis@acm.org, j.pitt@imperial.ac.uk

Abstract. We have been developing a framework for executable specification of norm-governed multi-agent systems. In this framework, specification is a design-time activity; moreover, there is no support for run-time modification of the specification. Due to environmental, social, or other conditions, however, it is often desirable, or even necessary, to alter the system specification during the system execution. In this chapter we extend our framework by allowing for 'dynamic specifications', that is, specifications that may be modified at run-time by the members of a system. The framework extension is motivated by Brewka's 'dynamic argument systems' — argument systems in which the rules of order may become the topic of the debate. We illustrate our framework for dynamic specifications by presenting: (i) a dynamic specification of an argumentation protocol, and (ii) an execution of this protocol in which the participating agents modify the protocol specification.

1 Introduction

A particular kind of Multi-Agent System (MAS) is one where the member agents are developed by different parties, and where there is no direct access to an agent's internal state. In this kind of MAS it cannot be assumed that all agents will behave according to the system specification because the agents act on behalf of parties with competing interests, and thus they may inadvertently fail to, or even deliberately choose not to, conform to the system specification in order to achieve their individual goals. A few examples of this type of MAS are Virtual Organisations, electronic marketplaces, argumentation (dispute resolution) protocols, and negotiation protocols. MAS of this type are often classified as 'open'.

We have been developing executable specifications of open MAS [1, 3]; we adopt a bird's eye view of these systems, as opposed to an agent's own perspective whereby it reasons about how it should act. Furthermore, we view agent systems as instances of *normative systems* [18]. A feature of this type of system is that actuality, what is the case, and ideality, what ought to be the case, do not necessarily coincide. Therefore, it is essential to specify what is permitted, prohibited, and obligatory, and perhaps other more complex normative relations

that may exist between the agents. Amongst these relations, we place considerable emphasis on the representation of *institutionalised power* [19] — a standard feature of any norm-governed system whereby designated agents, when acting in specified roles, are empowered by an institution to create specific relations or states of affairs (such as when an agent is empowered by an institution to award a contract and thereby create a bundle of normative relations between the contracting parties). We encode specifications of open MAS in executable action languages from the field of Artificial Intelligence [12, 20].

Our executable specifications may be classified as 'static', in the sense that there is no support for their run-time modification. In some open MAS, however, environmental, social or other conditions may favour, or even require, specifications modifiable during the system execution. Consider, for instance, the case of a malfunction of a large number of sensors in a sensor network, or the case of manipulation of a voting procedure due to strategic voting, or when an organisation conducts its business in an inefficient manner. Therefore, we present in this chapter an infrastructure for 'dynamic specifications', that is, specifications that are developed at design-time but may be modified at run-time by the members of a system. The presented infrastructure is motivated by Brewka's 'dynamic argument systems' [6] — argument systems in which, at any point in the disputation, participants may start a meta level debate, that is, the rules of order can become the current point of discussion, with the intention of altering these rules.

Our infrastructure for dynamic specifications allows protocol participants to alter the rules of a protocol P during the protocol execution. P is considered an 'object' protocol; at any point in time during the execution of the object protocol the participants may start a 'meta' protocol in order to decide whether the object protocol rules should be modified: add a new rule-set, delete an existing one, or replace an existing rule-set with a new one. Moreover, the participants of the meta protocol may initiate a meta-meta protocol to decide whether to modify the rules of the meta protocol, or they may initiate a meta-meta-meta protocol to modify the rules of the meta-meta protocol, and so on.

We chose an argumentation protocol based on Brewka's reconstruction of a theory of formal disputation to illustrate our infrastructure for dynamic specifications: the object and meta protocols are all argumentation protocols. In other words, at any time during a debate the agents may start a meta level argument to change the rules that govern their debate. The argumentation protocol was chosen for the sake of providing a concrete example. In general, the object protocol may be any protocol for open MAS, such as a protocol for resource-sharing, coordination or e-commerce; similarly a meta protocol can be any procedure for decision-making over rule modification (voting, negotiation, and so on). This issue is further discussed in the final section of the chapter.

The remainder of this chapter is organised as follows. First, we briefly review the Event Calculus, the action language that we employ to formalise system specifications. Second, we review our static specification of an argumentation protocol (extensively presented in [3]). Third, we present a dynamic specifica-

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

tion of the argumentation protocol and an infrastructure for modifying the argumentation protocol specification at run-time. Fourth, we present an execution of the protocol, demonstrating how the agents may alter the protocol specification. Finally, we compare our work to Brewka's account and to other research on dynamic protocol specifications, and outline directions for further research.

2 The Event Calculus

The Event Calculus (EC), introduced by Kowalski and Sergot [20], 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 following sections present a logic programming implementation of an EC action description expressing our argumentation protocol specification.

Table 2. Main Actions of RTFD*.

Action	Textual Description
$claim(Protag, Q)$	$Protag$ claims Q
$concede(Protag, Q)$	$Protag$ concedes to Q
$retract(Protag, Q)$	$Protag$ retracts Q
$deny(Protag, Q)$	$Protag$ denies Q
$declare(Det, Protag)$	Det declares $Protag$ the winner of the disputation
$objected(Ag)$	Ag objects to an action

3 An Argumentation Protocol

In this section we briefly present an argumentation (dispute resolution) protocol based on Brewka's account [6] of Rescher's theory of formal disputation [30]. A detailed description of this protocol, which we call RTFD*, and a formalisation in the action language $C+$ [12] can be found in [3]. In this chapter we present a formalisation of RTFD* in the Event Calculus because this action language has proven to be more appropriate for supporting run-time activities (see [1, 3] for a discussion about expressing protocol specifications in $C+$ and the Event Calculus).

There are three roles in the protocol: proponent, opponent and determiner. The protocol commences when the proponent claims the topic of the argumentation — any other action does not count as the commencement of the protocol. The protagonists (proponent and opponent) then take it in turn to perform actions, such as claiming, conceding to, retracting or denying a proposition. Each turn lasts for a specified time period during which the protagonist may perform several actions (send several messages) up to some specified limit. After each such action the other participants are given an opportunity to object within another specified time period. In other words, Ag's action Act is followed by a time period during which Ag may not perform any actions and the other participants may object to Act. The determiner may declare the winner only at the end of the argumentation, that is, when the specified period for the argumentation elapses. If at the end of the argumentation both the proponent and opponent have 'accepted' the topic of the argumentation (in a sense that will be made clear later), then the determiner may only declare the proponent the winner. If, however, the proponent does not accept the topic then the determiner may only declare the opponent the winner. Finally, if the proponent accepts the topic and the opponent does not, the determiner has *discretion* to declare either of them the winner. It may also have an *obligation* to decide one way or the other, depending on which version of the protocol we choose to adopt.

Table 2 displays the main actions of RTFD* whereas Table 3 presents a number of the fluents of the EC action description expressing the RTFD* specifica-

Table 3. Main Fluents of the RTFD* Specification.

Fluent	Domain	Textual Description
turn	{*proponent*, *opponent*, *determiner*}	the turn to 'speak'
role_of(*Ag*)	{*proponent*, *opponent*, *determiner*}	the role *Ag* occupies
premise(*Protag*, *Q*)	{t, u, f}	*Protag* has an explicit, unconfirmed, or no premise about *Q*
accepts(*Protag*, *Q*)	boolean	*Protag* accepts *Q*
objectionable(*Act*)	boolean	*Act* is objectionable
pow(*Ag*, *Act*)	boolean	*Ag* is empowered to perform *Act*
protocol	{*initial*, *executing*, *idle*}	the protocol is at the initial state, executing, or idle

tion. *Ag* is a variable expressing protocol participants, *Protag* expresses protocol protagonists, *Det* denotes the agent occupying the role of determiner, *Q* denotes the propositions that protagonists may claim, concede to, retract or deny, and *Act* represents a claim, concede, retract or deny action. The fluents of the EC action description are inertial. The semantics of the actions and the utility of the fluents will be explained in the following sections.

3.1 Physical Capability

The *system events* of the RTFD* specification are the timeouts — these are issued by a global clock. A type of timeout event is used to denote the turn of each participant. When RTFD* commences (this happens when the proponent claims the topic of the argumentation) a global clock starts 'ticking'. The first timeout signals the end of the proponent's turn and the beginning of the opponent's turn to 'speak', by setting *turn* = *opponent* (see Table 3 for a description of the *turn* fluent). The next timeout signals the end of the opponent's turn and the beginning of the proponent's turn, by setting *turn* = *proponent*, and so on.

The remaining actions of the RTFD* specification are those performed by the protocol participants (see Table 2). It is a feature of RTFD* that an agent may object to the actions of another participant. The action *objected*(*Ag*) represents that an objection has been made by agent *Ag*. We abstract away details of how an objection is transmitted within the specified deadline (recall that every action *Act* is followed by a time period during which no action may take place apart from an objection to *Act*). Instead, each 'step' of RTFD* corresponds to a claim, concede, retract, deny, or declare action by one of the participants together with an indication of whether that action was objected to by one or more of the other

participants. For example,

$$\begin{array}{l}\mathsf{happens}(\mathit{objectedClaim}(\mathit{Protag}, Q), T) \leftarrow \\ \quad \mathsf{happens}(\mathit{claim}(\mathit{Protag}, Q), T), \\ \quad \mathsf{happens}(\mathit{objected}(Ag), T)\end{array} \tag{1}$$

represents a claim that Q by *Protag* that has been objected to by some other participant. Similarly,

$$\begin{array}{l}\mathsf{happens}(\mathit{notObjectedClaim}(\mathit{Protag}, Q), T) \leftarrow \\ \quad \mathsf{happens}(\mathit{claim}(\mathit{Protag}, Q), T), \\ \quad \forall Ag\ \mathsf{not}\ \mathsf{happens}(\mathit{objected}(Ag), T)\end{array} \tag{2}$$

expresses a claim that has not been objected to ('not' denotes 'negation by failure' [8]). The object mechanism is beyond the scope of this chapter; for an extensive discussion about this issue see [3].

We have chosen to specify that any protagonist is always capable of signalling a claim, concede, retract, deny, and object action, and the determiner is always capable of signalling a declare and object action. The effects of these actions are presented next.

At the initial protocol state the protagonists have no premises, that is, the value of every $\mathit{premise}(\mathit{Protag}, Q)$ fluent is f. The protocol commences with the proponent's claim of the topic. The effects of a claim are expressed as follows:

$$\begin{array}{l}\mathsf{initiates}(\mathit{notObjectedClaim}(\mathit{Protag}, Q), \mathit{premise}(\mathit{Protag}, Q) = \mathsf{t}, T) \leftarrow \\ \quad \mathsf{holdsAt}(\mathit{premise}(\mathit{Protag}, Q) = \mathsf{f}, T)\end{array} \tag{3}$$

Rule (3) expresses that *Protag*'s claim of Q leads from a state in which *Protag* has no explicit premise that Q (that is, $\mathit{premise}(\mathit{Protag}, Q) = \mathsf{f}$) to a state in which it does have an explicit premise that Q (that is, $\mathit{premise}(\mathit{Protag}, Q) = \mathsf{t}$), on the condition that no (other) agent objects to the claim. An objection is only effective in blocking the effects of a claim if it (the objection) is well-founded (in a sense to be specified below). If the objection is not well-founded then it does not block the effects of the claim (though it might have other effects, such as exposing the objecting agent to sanctions). We therefore add the constraint:

$$\begin{array}{l}\mathsf{initiates}(\mathit{objectedClaim}(\mathit{Protag}, Q), \mathit{premise}(\mathit{Protag}, Q) = \mathsf{t}, T) \leftarrow \\ \quad \mathsf{holdsAt}(\mathit{objectionable}(\mathit{claim}(\mathit{Protag}, Q)) = \mathsf{false}, T), \\ \quad \mathsf{holdsAt}(\mathit{premise}(\mathit{Protag}, Q) = \mathsf{f}, T)\end{array} \tag{4}$$

Boolean fluents $\mathit{objectionable}(\mathit{Act})$ are used to represent that an objection to *Act* is well-founded.

Suppose that protagonist *Protag* claims a proposition Q. Opponent Protag' may respond to *Protag*'s claim by conceding to, or denying the claim. If Protag' does neither then we say that Protag' has an 'unconfirmed' premise that Q, denoted by $\mathit{premise}(\mathit{Protag}', Q) = \mathsf{u}$. The value of a *premise* fluent is set to 'un-

confirmed' as follows, for every pair of distinct protagonists $Protag$ and $Protag'$:

$$\begin{aligned}&\mathsf{initiates}(notObjectedClaim(Protag, Q), premise(Protag', Q) = \mathsf{u}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(premise(Protag, Q) = \mathsf{f}, T), \\ &\quad \mathsf{holdsAt}(premise(Protag', Q) = \mathsf{f}, T)\end{aligned} \tag{5}$$

$$\begin{aligned}&\mathsf{initiates}(objectedClaim(Protag, Q), premise(Protag', Q) = \mathsf{u}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(objectionable(claim(Protag, Q)) = \mathsf{false}, T), \\ &\quad \mathsf{holdsAt}(premise(Protag, Q) = \mathsf{f}, T), \\ &\quad \mathsf{holdsAt}(premise(Protag', Q) = \mathsf{f}, T)\end{aligned} \tag{6}$$

In other words, $Protag$'s claim of Q leads (subject to possible objections) to a state in which $Protag'$ has an unconfirmed premise that Q, provided that $Protag'$ does not already have a premise that Q (that is, provided that the value of $premise(Protag', Q)$ is f). If $Protag'$ already has a premise that Q (that is, $premise(Protag', Q) = \mathsf{t}$) then its premise does not become unconfirmed, and it does not need to respond to $Protag$'s claim.

A response to claim is a concession or a denial; consider the effects of a concession:

$$\begin{aligned}&\mathsf{initiates}(notObjectedConcede(Protag, Q), premise(Protag, Q) = \mathsf{t}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(premise(Protag, Q) = \mathsf{u}, T)\end{aligned} \tag{7}$$

$$\begin{aligned}&\mathsf{initiates}(objectedConcede(Protag, Q), premise(Protag, Q) = \mathsf{t}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(objectionable(concede(Protag, Q)) = \mathsf{false}, T), \\ &\quad \mathsf{holdsAt}(premise(Protag, Q) = \mathsf{u}, T)\end{aligned} \tag{8}$$

Similarly we may express the effects of a denial and a retraction in terms of the protagonists' premises. Regarding declarations, a $declare(Det, Protag)$ action signals $Protag$ the winner of the dispute (subject to objections). (For more details on the effects of the protocol actions see [3]).

We now turn our attention to objections: when is an objection to an action Act effective in blocking the effects of Act, that is, when is Act objectionable? When the agent that performed Act did not have the 'institutional power' to perform Act. An account of institutional power in the context of the argumentation protocol is given next.

3.2 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 [31], 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 [19], 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).

We use the concept of institutional power in the argumentation protocol specification as follows. We say that, for example, sending a $claim(Ag, Q)$ message while having the institutional power to make a claim, counts, in the argumentation protocol, as a non-objectionable claim, that is, a claim whose effects cannot be blocked. If, however, the claim is uttered by an agent without the power to make the claim, then this action will be objectionable and, therefore, its effects will be blocked by an objection issued by another agent. The same applies to the remaining protocol actions.

The institutional power to make a claim, for instance, is formalised as follows:

$$\begin{aligned} &\mathsf{holdsAt}(pow(Protag, claim(Protag, Q)) = \mathsf{true}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(premise(Protag, Q) = \mathsf{f}, T), \\ &\quad \mathsf{holdsAt}(protocol = executing, T), \\ &\quad \mathsf{holdsAt}(role\_of(Protag) = Role, T), \\ &\quad \mathsf{holdsAt}(turn = Role, T) \end{aligned} \tag{9}$$

According to the above rule, $Protag$ is empowered to claim Q if: (i) $Protag$ does not have a premise that Q, (ii) the protocol is 'executing', that is, it is neither in the initial state nor has it finished, and (iii) it is $Protag$'s turn to 'speak'. The power to make a claim at the initial protocol state, and, in general, to perform the remaining protocol actions, is formalised in a similar manner.

As mentioned above, performing an action without the corresponding institutional power constitutes this action objectionable; an objectionable claim, for instance, as defined as follows:

$$\begin{aligned} &\mathsf{holdsAt}(objectionable(claim(Protag, Q)) = \mathsf{true}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(pow(Protag, claim(Protag, Q)) = \mathsf{false}, T) \end{aligned} \tag{10}$$

Similarly we define when the remaining protocol actions are objectionable.

The specification of the procedural part of the argumentation includes a specification of permitted and obligatory actions, as well as sanctioning mechanisms to address the performance of forbidden actions and non-compliance with obligations — see [3]. We do not present here this aspect of the procedural part of the argumentation, and the logic of disputation (although a brief discussion about this logic will be presented later). The protocol specification presented so far is sufficient for illustrating the infrastructure for changing the protocol specifications at run-time, which is the aim of this chapter.

4 A Dynamic Argumentation Protocol

Being motivated by Brewka [6], we present an infrastructure that allows protocol participants to modify (a subset of) the rules of an argumentation protocol at

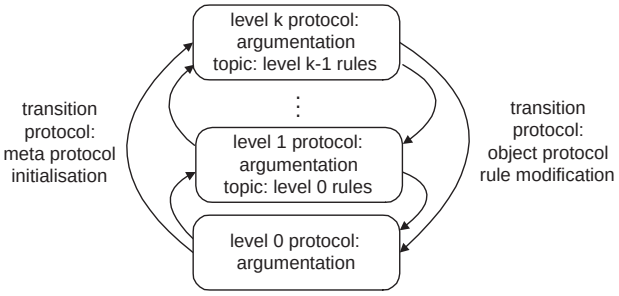

Fig. 1. An Infrastructure for Run-Time Protocol Modification.

run-time. More precisely, we consider the argumentation protocol as an 'object' protocol; at any point in time during the execution of the object protocol the participants may start a 'meta' argumentation protocol in order to potentially modify the object protocol rules: add a new rule-set, delete an existing one, or replace an existing rule-set with a new one. The topic of the dispute of the meta protocol is the proposed rule modification of the object protocol. Moreover, the participants of the meta protocol may initiate a meta-meta protocol to modify the rules of the meta protocol, or they may initiate a meta-meta-meta protocol to modify the rules of the meta-meta protocol, and so on — see Figure 1. In general, in a k-level infrastructure, level 0 corresponds to the main (argumentation) protocol while a protocol of level n, $0 < n \leq k$, is created, by the protocol participants of level m, $0 \leq m < n$, in order to debate over the protocol rules of level $n-1$.

Each protocol level has its own protocol state; for instance, agent Ag_1 may occupy the role of proponent in level 0 and the role of opponent in level 1 (role-assignment in meta protocols will be discussed in the following section), Ag_2 may be empowered to claim q in level 0 but have no powers in level 2, it may be the opponent's turn to 'speak' in level 1 and the determiner's turn to 'speak' in level 3, and so on. In order to distinguish between the protocol states of different protocol levels, we relativise the protocol rules according to the protocol level. More precisely, we add a parameter in the representation of actions and fluents, expressing the protocol level PL, as follows: $claim(Protag, Q, PL)$, $objected(Ag, PL)$, and so on for actions, $turn(PL)$, $objectionable(Act, PL)$, and so on for fluents.

The rules of an argumentation protocol are divided into two categories: 'core' rules, that is, rules that are always part of the protocol specification, and 'replaceable' rules, that is, rules that may be deleted, or replaced by other rules, under certain circumstances, during the protocol execution by means of a meta protocol. Consider, for example, the 'silence implies consent' property of the presented argumentation protocol. This property can be summarised as follows: a protagonist that does not explicitly respond to a claim by the other protagonist is assumed to concede to the claim. We may specify the rules expressing this

property as replaceable, that is, the protocol participants may choose to include this property in, or exclude it from, the protocol specification at any point in time during the protocol execution.

The rules that are part of a protocol at a given time are called 'active' (clearly, core rules are always active during a protocol execution) whereas all other rules are called 'inactive' (this is similar to what is called 'external time of norms' — see, for example, [22]). Protocols of different levels may not have the same set of active rules. For example, at a particular point in time t a replaceable rule-set tagged as 'sic', expressing the silence implies consent property, may be active in level 0 and level 2 but inactive in level 1; this is expressed as follows:

$$\mathsf{holdsAt}(\mathit{active}(\mathit{sic}) = [0, 2], t)$$

The $\mathit{active}(R)$ fluent expresses the protocol levels in which a rule-set R is active. To illustrate the meaning of the *active* fluent consider the following example:

$$\begin{aligned}
&\mathsf{holdsAt}(\mathit{accepts}(\mathit{Protag}, P, \mathit{PL}) = \mathsf{true}, T) \leftarrow \\
&\quad \mathsf{holdsAt}(\mathit{active}(\mathit{accept}) = \mathit{List}, T), \\
&\quad \mathit{PL} \in \mathit{List}, \\
&\quad \mathsf{holdsAt}(\mathit{premise}(\mathit{Protag}, Q, \mathit{PL}) = \mathsf{t}, T), \\
&\quad \mathit{implies}(Q, P)
\end{aligned} \tag{11}$$

$$\begin{aligned}
&\mathsf{holdsAt}(\mathit{accepts}(\mathit{Protag}, P, \mathit{PL}) = \mathsf{true}, T) \leftarrow \\
&\quad \mathsf{holdsAt}(\mathit{active}(\mathit{sic}) = \mathit{List}, T), \\
&\quad \mathit{PL} \in \mathit{List}, \\
&\quad (\ \mathsf{holdsAt}(\mathit{premise}(\mathit{Protag}, Q, \mathit{PL}) = \mathsf{t}, T)\ ; \\
&\qquad \mathsf{holdsAt}(\mathit{premise}(\mathit{Protag}, Q, \mathit{PL}) = \mathsf{u}, T)\ ), \\
&\quad \mathit{implies}(Q, P)
\end{aligned} \tag{11$'$}$$

Rules (11) and (11′) provide simple, alternative formalisations of the conditions in which a protagonist 'accepts' a proposition — the determiner declares the winner of the dispute according to what is accepted by both protagonists. The first two conditions of these rules express whether these rules are active in a protocol level. Rule (11) is tagged as 'accept'; according to this rule, *Protag* accepts P in a protocol of level PL if: (i) the 'accept' rule is active in level PL, (ii) *Protag* has an explicit premise that Q in level PL, and (iii) P is a (classical) logical implication of Q. The *implies* are simply suitably chosen atemporal predicates[3]. Rule (11′) incorporates the silence implies consent property: a protagonist accepts all logical implications of each of its explicit *and unconfirmed* premises — recall that $\mathit{premise}(\mathit{Protag}, Q, \mathit{PL}) = \mathsf{u}$ expresses that *Protag* has an unconfirmed premise that Q in a protocol of level PL, that is, *Protag* has not responded to a claim that Q made by the other protagonist in level PL. (';' expresses disjunction.) Given rules (11) and (11′), a protocol of level n exhibits the silence implies consent property if the rule tagged as 'sic' is active in this level;

[3] This is a very simple formalisation of the logic of disputation that suffices, however, for presenting a simple example; more complicated formalisations of this logic are out of the scope of this chapter.

if the 'accept' rule is active, on the other hand, this property is not incorporated in the protocol level.

Apart from adding, deleting and replacing rule-sets, protocol participants may also modify the values of certain components of the protocol specification. In the argumentation protocol example, participants may change at run-time the number of turns that each protagonist may take, and the duration of a turn.

In order to modify the protocol rules of level n (for example, to replace the 'accept' rule with the 'sic' rule), that is, in order to start a protocol of level $n+1$, the protocol participants of level n need to follow a 'transition protocol' — see Figure 1. We present an example specification of a transition protocol next.

4.1 Transition Protocol

For the sake of an example we have specified a very simple transition protocol: a protocol protagonist of level n proposes a modification of the rules of this protocol level (or of the rules of level $n+m$). If the protagonist is empowered to propose such a change then the protocol of level $n+1$ (or $n+m+1$) begins; otherwise the proposal is ignored. Clearly, in more realistic scenarios the transition protocol would be more complicated. For instance, a protagonist's proposal for rule modification should be seconded by another participant, say the determiner, before the meta protocol commences, or a protagonist's proposal could initiate a meta protocol even if the protagonist is not empowered to make the proposal, provided that no other agent objects. Moreover, a proposal for rule modification may not necessarily initiate a meta protocol — the proposal may be accepted by the participants resulting in the immediate application of the rule modification. Such transition protocols could be formalised similar to the argumentation protocol presented so far, or other interaction protocols presented elsewhere [1, Chapters 4 and 6].

In this simple example we have specified the power to propose a rule-set replacement as follows:

$$\begin{aligned}
&\mathsf{holdsAt}(pow(Ag, propose(Ag, replace(OldRule, NewRule), PL', PL)) = \mathsf{true}, T) \leftarrow \\
&\quad PL' \geq PL, \\
&\quad (\ \mathsf{holdsAt}(role_of(Ag, PL) = proponent, T)\ ; \\
&\quad\ \ \mathsf{holdsAt}(role_of(Ag, PL) = opponent, T)\ ), \\
&\quad \mathsf{holdsAt}(protocol(PL'+1) = idle, T), \\
&\quad \mathsf{holdsAt}(active(OldRule) = List, T), \\
&\quad PL' \in List, \\
&\quad \mathsf{holdsAt}(active(NewRule) = List_2, T), \\
&\quad PL' \notin List_2
\end{aligned} \tag{12}$$

The above rule states that a protocol participant Ag of level PL is empowered to propose to replace an $OldRule$ rule-set, from level PL', with a $NewRule$ one if:

- $PL' = PL$, that is, Ag proposes a rule modification in the protocol that currently participates, or $PL' > PL$, that is, Ag proposes a rule modification in a meta protocol.
- Ag occupies the role of proponent or that of opponent in level PL. In other words, in this example specification the determiner is not empowered to propose a modification of the protocol rules.
- There is no protocol taking place in level $PL'+1$. In this example, only one protocol may take place in each level. Therefore, an argumentation protocol for modifying the rules of level PL', that is, a protocol of level $PL'+1$, may commence only if there is no other protocol of level $PL'+1$ taking place.
- The *OldRule* rule-set is active in level PL'.
- The *NewRule* rule-set is inactive in level PL'.

The power to propose the addition or deletion of a rule-set is formalised in a similar manner.

In a more realistic scenario the power to propose a rule modification would probably have additional conditions. For instance, a rule-set should only be replaced by an 'interchangeable' rule-set (it should not be possible, for example, to replace a rule-set expressing the conditions in which an agent is permitted to perform an action with a rule-set expressing the conditions in which a proposition is accepted by a protagonist). In some systems an agent would not have the power to propose a rule modification (or the power to accept or second a rule modification, in more complex transition protocols) that would create a (type of) protocol inconsistency. In other systems an agent may be empowered to propose a rule modification that leads to a protocol inconsistency — the acceptance or not of the proposed modification would be decided based on the arguments of the proposing agent presented in the meta level. Moreover, inactive rule-sets proposed to become active may come from a specified rule library, thus allowing only for pre-determined protocol changes, or agents may propose the addition of completely new rule-sets. (Vreeswijk [33], for instance, allows for pre-determined protocol changes whereas Brewka [6] allows for any type of protocol change. There is no comment, however, in the latter approach on the issue of creating protocol inconsistencies.) The formalisation of examples taking under consideration these issues is an area of current research.

The effect of a 'successful' proposal for rule modification in level n, that is, the effect of proposing a rule modification in level n while having the power to make the proposal, is the initiation of a protocol of level $n+1$. The topic of the latter protocol is the proposed rule modification (for example, which rule-set should be replaced by which), the agent that made the proposal occupies the role of proponent, the other protagonist occupies the role of opponent, and the determiner remains the same. The fact that an agent may successfully start a protocol of level $n+1$ by proposing a modification of the protocol rules of level n, however, does not necessarily imply that the rules of level n will be modified. It is only if the agent that successfully proposed the modification is declared the winner of the argument of level $n+1$ that the rules of level n will be modified. Consider the following rules expressing the outcome of a protocol of level $n+1$

that took place in order to replace the *OldRule* rule-set with the *NewRule* rule-set in level n:

$$\begin{aligned}
&\mathsf{initiates}(endTimeout(PL), active(NewRule) = NewList, T) \leftarrow \\
&\quad \mathsf{holdsAt}(topic(PL) = replace(OldRule, NewRule), T), \\
&\quad \mathsf{holdsAt}(winner(PL) = Winner, T), \\
&\quad \mathsf{holdsAt}(role_of(Winner, PL) = proponent, T), \\
&\quad \mathsf{holdsAt}(active(NewRule) = List, T), \\
&\quad NewList := List \cup \{PL{-}1\}
\end{aligned} \tag{13}$$

$$\begin{aligned}
&\mathsf{initiates}(endTimeout(PL), active(OldRule) = NewList, T) \leftarrow \\
&\quad \mathsf{holdsAt}(topic(PL) = replace(OldRule, NewRule), T), \\
&\quad \mathsf{holdsAt}(winner(PL) = Winner, T), \\
&\quad \mathsf{holdsAt}(role_of(Winner, PL) = proponent, T), \\
&\quad \mathsf{holdsAt}(active(OldRule) = List, T), \\
&\quad NewList := List \setminus \{PL{-}1\}
\end{aligned} \tag{14}$$

Rule (13) states that an $endTimeout(PL)$, that is, the last timeout of level PL, results in activating the *NewRule* rule-set in level $PL{-}1$ if: (i) the topic of the argument of level PL was the replacement of the *OldRule* rule-set with the *NewRule* one, and (ii) the winner of the argument of level PL was the agent occupying the role of proponent. If the winner was the opponent or there was no declared winner, then the *NewRule* rule-set would not have been activated in level $PL{-}1$. Similarly, rule (14) states that an $endTimeout(PL)$ results in making the *OldRule* rule-set inactive in level $PL{-}1$ if the two aforementioned conditions hold.

5 Animation

In order to illustrate the proposed infrastructure for dynamic specifications, in this section we animate an example run of a 2-level argument system. A part of the narrative of events of this run is displayed in Table 4 ($oTimeout(PL)$ denotes a timeout initiating the opponent's turn to 'speak' in level PL; similarly $pTimeout(PL)$ and $dTimeout(PL)$ denote, respectively, timeouts initiating the proponent's and the determiner's turn to 'speak' in level PL). This narrative is motivated by Brewka's case study of a dynamic argument system [6, Section 7]. In the example presented here the argumentation protocol includes a single replaceable component, rule (11′), expressing that the propositions a protagonist accepts are determined by the silence implies consent property (see Section 4). Rule (11′) is initially active in all protocol levels.

Given that the RTFD* specification is expressed as a logic program, we may query our implementation to determine the system state current at each time and protocol level (for instance, which roles each participant occupies, what premises each protagonist has, what powers each participant has, which actions are objectionable, and so on). We will discuss next the system states of the run displayed in Table 4.

Table 4. A Sample Run of RTFD*

Time	Action
0	$claim(agent_1, murderer(jack), 0)$
	...
14	$claim(agent_1, on(blood, shoe), 0)$
14	$objected(agent_2, 0)$
15	$oTimeout(0)$
18	$claim(agent_2, illegal\_info(on(blood, shoe)), 0)$
30	$pTimeout(0)$
31	$concede(agent_1, illegal\_info(on(blood, shoe)), 0)$
45	$oTimeout(0)$
46	$propose(agent_2, replace(sic, sic\_ill\_info), 0, 0)$
49	$claim(agent_2, replace(sic, sic\_ill\_info), 1)$
	...
77	$dTimeout(1)$
78	$declare(det, agent_2, 1)$
93	$endTimeout(1)$
	...
135	$endTimeout(0)$

The protocol of level 0 commences with the claim of $agent_1$, occupying the role of proponent, that Jack is a murderer, that is, the topic of the argument of level 0 — see time-point 0 in Table 4 (recall that the last parameter of an action is the protocol level in which the action is performed). At time-point 14 $agent_1$ makes a claim that there is evidence, victim's blood on Jack's shoe, that convicts Jack as the murderer. The evidence, however, was obtained illegally. Therefore, at the same time $agent_2$, occupying the role of opponent, objects to $agent_1$'s claim. $agent_2$'s objection is unsuccessful in blocking the effects of $agent_1$'s claim because the claim is not objectionable: $agent_1$ is empowered to make the claim at that time since it does not have an explicit premise that there is victim's blood on Jack's shoe, the protocol is executing, and it is $agent_1$'s turn to 'speak' (see rule (9)). The effects of $agent_1$'s claim are that $premise(agent_1, on(blood, shoe), 0) = \mathsf{t}$ and $premise(agent_2, on(blood, shoe), 0) = \mathsf{u}$ (see rules (4) and (6)). Consequently, according to rule (11′), currently active in level 0, expressing the conditions in which a proposition is accepted, both protagonists accept $on(blood, shoe)$. At time-point 15 a timeout takes place initiating $agent_2$'s turn to 'speak'. $agent_2$ does not deny $agent_1$'s claim that there is victim's blood on Jack's shoe; however, $agent_2$ claims that the presented evidence was obtained illegally. $agent_1$ concedes to this claim when it is its turn to 'speak' (at time-point 31) since, according to rule (11′), $agent_1$'s concession does not change the fact that $on(blood, shoe)$ is accepted by both protagonists.

In order to change what is accepted, $agent_2$ proposes a modification of the rules of level 0; it proposes to replace rule (11′) tagged as 'sic' with the rule

below, tagged as 'sic_ill_info':

$$\begin{aligned}
&\mathsf{holdsAt}(\mathit{accepts}(\mathit{Protag}, P, \mathit{PL}) = \mathsf{true}, T) \leftarrow \\
&\quad \mathsf{holdsAt}(\mathit{active}(\mathit{sic_ill_info}) = \mathit{List}, T), \\
&\quad \mathit{PL} \in \mathit{List}, \\
&\quad (\ \mathsf{holdsAt}(\mathit{premise}(\mathit{Protag}, Q, \mathit{PL}) = \mathsf{t}, T)\ ; \\
&\quad\ \ \mathsf{holdsAt}(\mathit{premise}(\mathit{Protag}, Q, \mathit{PL}) = \mathsf{u}, T)\ ), \\
&\quad \mathsf{holdsAt}(\mathit{illegal_info}(Q) = \mathsf{false}, T), \\
&\quad \mathit{implies}(Q, P)
\end{aligned} \tag{11''}$$

Rule (11″) is a variation of the silence implies consent property that considers illegally obtained evidence: *Protag* accepts evidence and its implications, put forward by itself or by the other protagonist *Protag′* (and not challenged by *Protag*), provided that the evidence is not illegally obtained. *illegal_info* are suitably chosen fluents.

$agent_2$'s proposal, at time 46, for modifying the rules of level 0 is 'successful' because $agent_2$ is empowered to make the proposal at that time (see rule (12)): the modification concerns the protocol level in which $agent_2$ currently participates, $agent_2$ occupies the role of opponent in level 0, the 'sic' rule is active and the 'sic_ill_info' rule is inactive in that level. The result of the 'successful' proposal is the commencement of a protocol of level 1, in which $agent_2$ occupies the role of proponent (since it initiated this protocol), $agent_1$ occupies the role of opponent, and the determiner remains the same. (Notice that, at the same time, $agent_2$ occupies the role of opponent and $agent_1$ that of proponent in level 0.) At time-point 49 $agent_2$ claims the topic of the protocol of level 1 which is the replacement of the 'sic' rule with the 'sic_ill_info' one. Following the argument in level 1 (on the modification of the rules of level 0), the determiner of level 1 declares $agent_2$ the winner; since $agent_2$ is the proponent of level 1 the proposed rule modification is applied in level 0, that is, the 'sic_ill_info' rule becomes active and the 'sic' one becomes inactive in level 0 (see rules (13) and (14)). (Notice, however, that the 'sic' rule is still active in level 1.) As a result, at time-point 94, after the last timeout of level 1, the values of the fluents $\mathit{accepts}(\mathit{agent}_1, \mathit{on}(\mathit{blood}, \mathit{shoe}), 0)$ and $\mathit{accepts}(\mathit{agent}_2, \mathit{on}(\mathit{blood}, \mathit{shoe}), 0)$ are not true, that is, no protagonist accepts that there is victim's blood on Jack's shoe.

It should be noted that the example presented above, in which the rule modification had retroactive effects, was chosen simply for illustration purposes. Clearly, there exist other examples in which the effects of a rule modification need to be applied only after the modification. Both alternatives may be expressed with the use of the Event Calculus.

In Brewka's example argument system [6, Section 7] the protocol of level 0 is modified in the following way in order to deal with illegal evidence: every action performed by a protagonist that has put forward illegally obtained evidence is considered objectionable as long as the protagonist does not retract the illegal evidence. This constraint may be incorporated in our formalisation by modifying our specication of objectionable actions. In general, there are several ways to deal with illegal evidence; the presented formalisation is but one example.

6 Conclusions and Future Research Directions

We presented a specification of an argumentation protocol in which the logic of disputation/argumentation (from which inferences are made to determine the winner) and the procedural part of the argumentation (that defines the conditions in which an agent is empowered, permitted, obliged to perform an action) were separated out from the mechanism by which either the argumentation logic or the argumentation procedure can be changed.

In the development of the static specification of the argumentation protocol [3] we were mainly concerned with the formalisation of the procedural part of the argumentation. Like Brewka [6] and [13, 14, 26, 28] we adopted a 'public protocol semantics' [29, Section 6], that is, we made no assumptions about the participants' internal architectures. We refined Brewka's distinction of possible and legal actions — we distinguished between physical capability, institutional power, and permission (and obligation). Moreover, unlike Brewka, we have been concerned with the *execution* of our protocol specifications. Indeed, in Section 5 we presented an execution of an example protocol specification; protocol participants may query at any point in time our executable specification in order to inform their decision-making. A more detailed comparison of our work with Brewka's account and related research from the argumentation field, from the standpoint of *static* argument systems, however, may be found in [3].

In this chapter we were concerned with the run-time modification of the protocol specification. As Brewka [6] points out, the need to allow for argumentation protocol rule modification at run-time by means of argumentation has already been identified in the literature (see, for example, [21]). A main difference of our work from Brewka's *dynamic* argument systems lies in the fact that we place emphasis on the transition protocol that is used to move from an object protocol to a meta protocol. We provided a concrete formalisation of a simple example transition protocol, distinguishing between successful and unsuccessful attempts to initiate a meta protocol. Moreover, we outlined alternative formalisations of more complex transition protocols and formalised a simple procedure for role-assignment in a meta level. In Brewka's simple example of a dynamic argument system a protocol participant may always successfully initiate a meta protocol; furthermore, there is no discussion about role-assignment in the meta level.

Vreeswijk [33] has also investigated forms of meta argumentation. The starting point for this work was two basic observations. Firstly, that there are different protocols appropriate for different contexts (for example, quick and shallow reasoning when time is a constraint; restricted number of counter-arguments when there are many rules and cases; etc). Secondly, that 'points of order', by which a participant may steer the protocol to a desired direction, are standard practice in dispute resolution meetings. Vreeswijk then defined a formal protocol for disputes in which points of order can be raised to allow (partial) protocol changes to be debated. A successful 'defence' meant that the parties in the dispute agreed to adopt a change in the protocol, and the rules of dispute were correspondingly changed.

As already mentioned in Section 4.1, Vreeswijk allowed only for pre-determined protocol changes. Unlike our work (and Brewka's work), meta argumentation was restricted to a single meta level. Moreover, there was no treatment of a 'transition protocol', that is, there was no formalisation of a procedure with which a participant could (attempt to) initiate a meta argument.

The example presented in our chapter included a single replaceable component — a 'partial protocol specification' in the terminology of Vreeswijk [33]. We could have formalised as replaceable other types of rule-set, such as the rule-sets expressing objectionable actions, permitted actions, obligatory actions, and so on. The decision concerning the classification of a rule-set as replaceable or core is application-specific.

When deciding to classify a rule-set as replaceable, one should consider the possible effects of making this rule-set inactive. As discussed in Section 4.1, the replacement of a rule-set with another one may create a type of protocol inconsistency — for instance, objecting to an objectionable action may no longer block the effects of this action, a participant may be forbidden and obliged to perform an action, and so on. Moreover, it may be required that a rule modification respects a set of protocol properties, such as 'soundness' and 'fairness' (see, for example, [24, 27, 33] for definitions of argumentation protocol properties). A way of verifying the effects of rule modification, by means of proving protocol properties, can be found in [3, Section 8].

The infrastructure presented here for dynamic specifications included an argumentation protocol in each level (see Figure 1). In principle, any protocol of level n, $n > 0$, could be any procedure for deciding whether or not to apply a rule modification (the protocol of level 0 is the main (argumentation) protocol). We could have, for instance, an infrastructure for dynamic argumentation protocol specifications in which some or all n level protocols ($n > 0$) are voting protocols, that is, agents take a vote on, instead of debating about, a proposed rule modification. The realisation of such an infrastructure (see [25] for a preliminary voting protocol formalisation), and in particular, the formalisation of a transition protocol leading from an argumentation protocol to a voting protocol, is an area of current work.

Apart from replacing an argumentation protocol of level n ($n > 0$) with a voting protocol, say, one could even replace the main argumentation protocol, that is, the protocol of level 0, with any type of protocol for open MAS (resource-sharing, negotiation, coordination, and so on). In this way it would be possible to have an infrastructure for dynamic resource-sharing protocol specifications, for instance. Such a setting requires suitable transition protocols that lead from level 0 (resource-sharing, for example) to level n, $n > 0$ (argumentation or voting).

On the topic of dynamic specifications for MAS, Bou and colleagues [4, 5] have presented a mechanism for the run-time modification of the norms of an 'electronic institution'. These researchers have proposed a 'normative transition function' that maps a set of norms (and goals) into a new set of norms: changing a norm requires changing its parameters, or its effect, or both. The 'institutional agents', representing the institution, are observing the members' interactions in

order to learn, with the use of case based reasoning, the normative transition function, so that the norms that will enable the achievement of the 'institutional goals' in a given scenario will be enacted.

Unlike Bou and colleagues, we do not necessarily rely on specific agents to (learn and) apply the modification of norms. In our case, any agent may (attempt to) adapt the system specification via meta protocols (argumentation, voting, negotiation or some other protocol). This does not exclude the possibility, however, that, in some applications, specific agents are given the institutional power to directly modify the system specification (without argumentation, voting, etc).

Chopra and Singh [7] present a way of adapting protocols according to context, or the preferences of agents in a given context. They formalise, in the action language $C+$ [12], protocols and 'transformers', that is, additions/enhancements to an existing protocol specification that handle some aspect of context or preference. Depending on the context or preference, a protocol specification is complemented, at *design-time*, by the appropriate transformer thus resulting in a new specification.

Like Chopra and Singh, we have used the action language $C+$ to express protocol specifications for open MAS (see, for instance, [1–3]). Unlike these researchers, we are concerned here with the *run-time* adaptation of a protocol specification and, therefore, we have developed an infrastructure (meta protocols, transition protocols) to achieve that.

Chopra and Singh, and Bou and colleagues, express protocols in terms of 'commitments' or obligations (here the term 'commitment' refers to a form of (directed) obligation between agents, and is *not* used as an alternative term for 'premise'). It is difficult to see how an interaction protocol for open MAS can be specified simply in terms of commitments in this sense. At the very least, a specification of a protocol's constitutive norms is also required.

The OMACS (Organisational Model for Adaptive Computational Systems) model [10, 11] is another approach for dynamic MAS. In OMACS, an agent-based organisation is represented by sets of agents, goals, roles, capabilities, and constraints, an assignment of goals and roles to agents, and a set of evaluative functions (for instance, *achieves*, *capable*, *possesses* and *potential*), that define how well a role achieves a goal, how well an agent can play a role, and so on. Given a trigger event, such as a change in the agents' capabilities, agents employ pre-compiled strategies or on-the-fly computed strategies, to adapt various aspects of an organisation, such as the assignment of roles to agents. The aim of adaptation is the maximisation of the value of the *organisation assignment function* that expresses the quality/efficiency of an organisation.

OMACS concentrates mainly on re-organisation from the perspective of a functional assignment — the assignment of goals and roles to agents — whereas we emphasise, like Bou and colleagues, a different perspective, aimed at a achieving a mapping form norms to norms, that is, the rules which regulate, among other things, the process of performing such a functional assignment. The outcome of the mapping still needs to be evaluated, though, as re-organisation is evaluated in OMACS. One way of evaluating our dynamic specifications is by ex-

amining whether certain protocol properties (such as the ones mentioned earlier) hold. Another way is discussed at the end of this section.

Identifying *when* to adapt a system specification during the system execution, as done in OMACS, for instance, with respect to agent availability, preferences, goals, capabilities, and so on, and in the work of Bou and colleagues with respect to the 'institutional goals', is a fundamental requirement for adaptive systems. This requirement has not been addressed by our work.

We have been concerned with a particular aspect of 'adaptation': the run-time modification of the 'rules of the game' of norm-governed systems. Clearly, there are other aspects of adaptive/dynamic systems, such as, for instance, the run-time modification of the (trading, and other) relationships between agents, the members of a system, the assignment of roles to agents (as done, for instance, in OMACS), and the goals of a system. [9, 15–17, 23, 32] are but a few examples of studies of adaptive systems.

To aid the evaluation process of a system with dynamic specifications, we have been working towards the development of a model, based upon the mathematical theory of Metric Spaces, that allows for quantification of a system's 'degrees of freedom', that is, the replaceable rule-sets. Use of the model allows for a human or machine evaluator to evaluate a system by considering the distance of the system's 'specification position', calculated with the use of the system's degrees of freedom, to a given point in the specification space (say, a desirable specification point or subspace). The distance is calculated with the use of a metric function. In addition to the model, we are developing software tools that largely automate the application of the model on a given system.

References

1. A. Artikis. *Executable Specification of Open Norm-Governed Computational Systems*. PhD thesis, University of London, November 2003. Retrieved January 8, 2008, from `http://www.iit.demokritos.gr/~a.artikis/publications/artikis-phd.pdf`, also available from the author.
2. A. Artikis, M. Sergot, and J. Pitt. Specifying electronic societies with the Causal Calculator. In F. Giunchiglia, J. Odell, and G. Weiss, editors, *Proceedings of Workshop on Agent-Oriented Software Engineering III (AOSE)*, LNCS 2585, pages 1–15. Springer, 2003.
3. A. Artikis, M. Sergot, and J. Pitt. An executable specification of a formal argumentation protocol. *Artificial Intelligence*, 171(10–15):776–804, 2007.
4. E. Bou, M. López-Sánchez, and J. Rodriguez-Aguilar. Towards self-configuration in autonomic electronic institutions. In *Proceedings of Workshop on Coordination, Organization, Institutions and Norms in agent systems*, LNCS 4386, pages 220–235. Springer, 2007.
5. E. Bou, M. López-Sánchez, and J. Rodriguez-Aguilar. Using case-based reasoning in autonomic electronic institutions. In *Proceedings of Workshop on Coordination, Organization, Institutions and Norms in agent systems*, LNCS 4870, pages 125–138. Springer, 2008.
6. G. Brewka. Dynamic argument systems: a formal model of argumentation processes based on situation calculus. *Journal of Logic and Computation*, 11(2):257–282, 2001.

7. A. Chopra and M. Singh. Contextualizing commitment protocols. In *Proceedings of Conference on Autonous Agents and Multi-Agent Systems (AAMAS)*, pages 1345–1352. ACM, 2006.
8. K. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 293–322. Plenum Press, 1978.
9. A. Rocha Costa and G. Pereira Dimuro. A minimal dynamical MAS organisation model. To appear in this volume, 2008.
10. S. DeLoach. OMACS: A framework for adaptive, complex systems. To appear in this volume, 2008.
11. S. DeLoach, W. Oyenan, and E. Matson. A capabilities-based model for adaptive organizations. *Autonomous Agents and Multi-Agent Systems*, 16(1):13–56, 2008.
12. E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1–2):49–104, 2004.
13. T. Gordon. The pleadings game: an exercise in computational dialectics. *Artificial Intelligence and Law*, 2:239–292, 1994.
14. T. Gordon. *The Pleadings Game: An Artificial Intelligence Model of Procedural Justice*. Kluwer Academic Publishers, 1995.
15. M. Hoogendoorn. Adaptation of organizational models for multi-agent systems based on max flow networks. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1321–1326, 2007.
16. M. Hoogendoorn, C. Jonker, M. Schut, and J. Treur. Modeling centralized organization of organizational change. *Computational & Mathematical Organization Theory*, 13(2):147–184, 2007.
17. B. Horling, B. Benyo, and V. Lesser. Using self-diagnosis to adapt organizational structures. In J. Müller, E. Andre, S. Sen, and C. Frasson, editors, *Proceedings of Conference on Autonomous Agents*, pages 529–536. ACM Press, 2001.
18. 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.
19. A. Jones and M. Sergot. A formal characterisation of institutionalised power. *Journal of the IGPL*, 4(3):429–445, 1996.
20. R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–96, 1986.
21. R. Loui. Process and policy: Resource-bounded non-demonstrative argument. Technical report, Washington University, Department of Computer Science, 1992.
22. R. Marín and G. Sartor. Time and norms: a formalisation in the event calculus. In *Proceedings of Conference on Artificial Intelligence and Law (ICAIL)*, pages 90–100. ACM Press, 1999.
23. C. Martin and K. S. Barber. Adaptive decision-making frameworks for dynamic multi-agent organizational change. *Autonomous Agents and Multi-Agent Systems*, 13(3):391–428, 2006.
24. P. McBurney, S. Parsons, and M. Wooldridge. Desiderata for agent argumentation protocols. In C. Castelfranchi and L. Johnson, editors, *Proceedings of Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 402–409. ACM Press, 2002.
25. J. Pitt, L. Kamara, M. Sergot, and A. Artikis. Voting in multi-agent systems. *Computer Journal*, 49(2):156–170, 2006.
26. H. Prakken. On dialogue systems with speech acts, arguments, and counterarguments. In *Proceedings of Workshop on Logics in Artificial Intelligence*, LNAI 1919, pages 224–238. Springer, 2000.

27. H. Prakken. Relating protocols for dynamic dispute with logics for defeasible argumentation. *Synthese*, 127(1–2):187–219, 2001.
28. H. Prakken. Coherence and flexibility in dialogue games for argumentation. *Journal of Logic and Computation*, 15:1009–1040, 2005.
29. H. Prakken. Formal systems for persuasion dialogue. *Knowledge Engineering Review*, 21(2):163–188, 2006.
30. N. Rescher. *Dialectics: A Controversy-Oriented Approach to the Theory of Knowledge*. State University of New York Press, 1977.
31. J. Searle. What is a speech act? In A. Martinich, editor, *Philosophy of Language*, pages 130–140. Oxford University Press, third edition, 1996.
32. Y. Shoham and M. Tennenholtz. On the emergence of social conventions: modeling, analysis and simulations. *Artificial Intelligence*, 94(1-2):139–166, 1997.
33. G. Vreeswijk. Representation of formal dispute with a standing order. *Artificial Intelligence and Law*, 8(2/3):205–231, 2000.