Executable Specification of Open Multi-Agent Systems

Alexander Artikis[12] and Marek Sergot[3]
[1]*Institute of Informatics and Telecommunications,*
National Centre for Scientific Research "Demokritos",
Athens 15310, Greece
[2]*Electrical & Electronic Engineering Department,*
Imperial College London, SW7 2BT, UK
[3]*Department of Computing,*
Imperial College London, SW7 2BZ, UK
E-mail: a.artikis@acm.org, mjs@doc.ic.ac.uk

Abstract. Multi-agent systems where the agents are developed by parties with competing interests, and where there is no access to an agent's internal state, are often classified as 'open'. The members of such systems may inadvertently fail to, or even deliberately choose not to, conform to the system specification. Consequently, it is necessary to specify the normative relations that may exist between the members, such as permission, obligation, and institutional power. We present a framework being developed for executable specification of open multi-agent systems. We adopt a bird's eye view of these systems, as opposed to an agent's perspective whereby it reasons about how it should act. This paper is devoted to the presentation of various examples from the NetBill protocol formalised in terms of institutional power, permission and obligation. We express the system specification in the Event Calculus and execute the specification by means of a logic programming implementation. We also give several example formalisations of sanctions for dealing with violations of permissions and obligations. We distinguish between an open multi-agent system and the procedure by which an agent enters and leaves the system. We present examples from the specification of a role-management protocol for NetBill, and demonstrate the interplay between such a protocol and the corresponding multi-agent system.

1. Introduction

A multi-agent system (MAS) is often classified as 'open' when there is no guarantee of benevolent behaviour and there is no access to an agent's internal state. Examples of open MAS of this sort are electronic markets, virtual organisations and digital right management applications. It has been argued that many practical applications in the future will be realised in terms of open MAS. Not surprisingly, there is a growing interest in the MAS community in these systems ([76, 54, 84, 69, 71, 85, 17] are but a few examples). Rosenschein and Zlotkin give the following description of this type of MAS:

> "The agents that we are interested in looking at are heterogeneous, self-motivated agents. [..] Agents do not necessarily have a notion of global utility. Each agent operating from your machine is interested in what your idea of utility is and in how to further your notion of goodness. [..] Agents do not act benevolently unless it's in their interest to do so. They do not necessarily share information, they do not necessarily do things that other agents ask them to do unless they have a good reason for doing so." [59, p.354]

We here consider a MAS as open if it exhibits the following characteristics:

- The internal architectures of the members are not publicly known.
- Members do not necessarily share a notion of global utility.
- The behaviour and the interactions of the members cannot be predicted in advance.

The first of these characteristics implies that an open MAS may be composed of agents with different internal architectures. Therefore, we will treat open MAS as *heterogeneous* ones. Moreover, there is no direct access to an agent's mental state and so we can only make inferences about that state. The second characteristic implies that the members of an open MAS may fail to, or even choose not to, conform to the system specifications in order to achieve their individual goals (this is what Minsky and Ungureanu [48] referred to as *inadvertent* and *malicious* violations respectively). And further, open MAS are always subject to unanticipated outcomes in their interactions [34].

Often in the literature open MAS are those where agents may enter or leave the system at any time (see, for example, [76,84,34,58]). Usually, agents enter or leave a system by participating in a *role-management protocol*, that prescribes the ways for applying for membership, withdrawing from a system, and so on. The specification of a role-management protocol is application-specific. Our definition of a MAS as open is irrespective of the protocols that specify the ways by which agents enter or leave the system.

In this paper we are concerned with presenting *executable specifications* of open MAS. 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. Moreover, we view open MAS as instances of *normative systems* [37]. 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, we specify what is permitted, prohibited and obligatory. Moreover, we explicitly represent the *institutional powers* [61, 38] of the agents, and maintain the standard, long established distinction between institutional power, permission and physical capability. Institutional power refers to the standard feature of any normative system whereby designated agents, when acting in specified roles, are empowered by an institution to create relations or states of affairs of special significance within the institution (such as when an agent is empowered by an institution to award a contract and thereby create a set of normative relations between the contracting parties).

In order to illustrate the concept of institutional power and the distinction between institutional power, permission and physical capability, this paper is devoted to the presentation of various examples formalised in terms of these concepts. In previous publications we were concerned with other aspects of normative systems; in [2] we presented a comparison of action formalisms for open MAS specification while in [1] we presented a specification of a dispute resolution protocol. In these previous papers we discussed the process of role-management but did not formalise it. Here we present an example specification of a protocol for managing applications for membership in an open MAS, withdrawal, etc, and demonstrate the interplay between such a protocol and the corresponding MAS.

We encode specifications of open MAS in executable action languages. In this paper we employ the Event Calculus (EC) [40], a simple and flexible formalism that is very easily and efficiently implemented for an important class of computational tasks. It thus provides a practical means of implementing an executable system specification.

The remainder of this paper is organised as follows. First, we present an example that we will use throughout the paper in order to illustrate the way we develop executable specifications of open MAS. This example is a variation of the NetBill protocol [72] that is used for buying and selling digital information on the Internet. Second, we describe the version of the Event Calculus that we employ to specify NetBill. Third, we present our specification of NetBill, which is a formalisation of the *social constraints* (or social laws) of NetBill expressed in terms of the institutional powers, permissions and obligations of the participants, as well as the sanctions that may be applied in order to deal with the violation of the social constraints. Fourth, we present an example specification of a role-management protocol for NetBill. In the penultimate

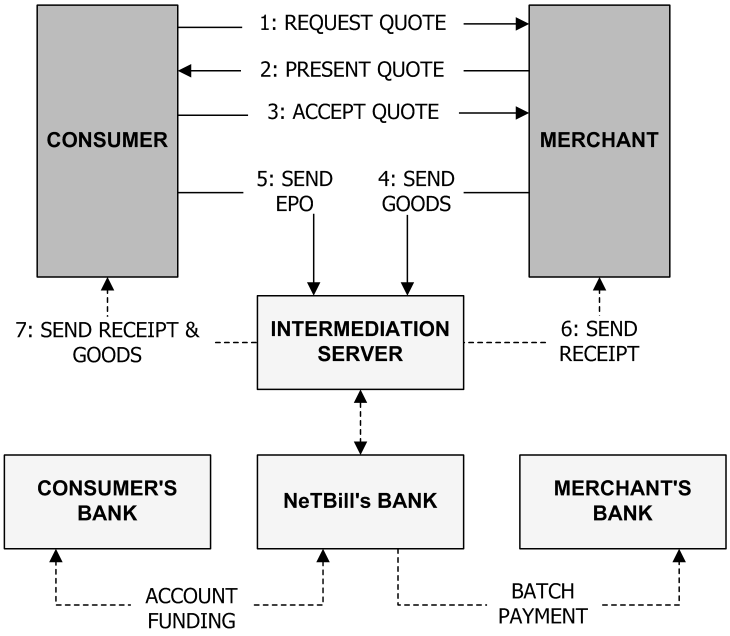

Figure 1. A Variation of the NetBill Payment Protocol.

section of the paper we present an execution of the specifications of NetBill and the associated role-management protocol. Finally, we compare our work to related approaches in the literature, and outline directions for further research.

2. Running Example: The NetBill Protocol

We employ a variation of the NetBill protocol [72] in order to demonstrate the way we construct executable specifications of open MAS. Figure 1 shows the participants and a run of the protocol. Initially, the consumer requests a quote for a set of goods and the merchant replies with the quote. Then, the consumer accepts the quote. Following the quote acceptance, the merchant sends the goods to an Intermediation Server (IS), while the consumer constructs an Electronic Payment Order (EPO) describing the transaction, and sends it to IS. Finally, IS takes care of the transfer of funds and sends a receipt to the merchant and the consumer — the consumer's receipt is accompanied by the merchant's goods.

IS acts as a trusted third party; it may 'cancel' a transaction, that is, refrain from transferring the funds to the merchant's account and sending the goods to the consumer, if the consumer's EPO does not clear or the merchant's goods are not of the agreed quality.

Clearly NetBill is an open MAS; it is composed of heterogeneous agents that have conflicting goals and act competitively. Actuality may not coincide with ideality: a merchant may over-advertise its goods, a consumer's EPO may not clear, and so on. Therefore, it is necessary to specify the rules expressing what constitutes ideal behaviour, and what are the consequences of deviating from such behaviour. The rules must define additional facts of special significance to NetBill, such as when a contract is established between two parties, when an agent is eligible to participate in NetBill, and so on. In the following sections we will present an example specification of NetBill addressing these issues.

To simplify the presentation and to keep the example manageable, we omit detailed discussion of the procedure that is followed to transfer the funds to the merchant's account, and the actions of IS in general (see the dotted lines in Figure 1). In the following sections we will focus on the consumers' and the merchants' actions, which are represented in Figure 1 by the continuous lines.

3. The Event Calculus

We have been using three action languages with direct routes to implementation to express protocol specifications:

1. The Event Calculus (EC) [40], a formal, intuitive and well-studied action language (see [2] for an EC specification of a contract-net protocol).

2. The $C+$ language [31], a formalism with an explicit transition systems semantics (see [1] for a $C+$ specification of a dispute resolution protocol).

3. The $nC+$ language [64, 13], an extended form of $C+$ designed specifically for representing simple normative and institutional concepts (see [63] for a $nC+$ specification of a simple resource sharing protocol).

Each formalism has its advantages and disadvantages for open MAS specifications. (See [2] for a comparison of their relative strengths.) 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 $nC+$ implementation (employing 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.)

EC, introduced by Kowalski and Sergot [40], 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 can 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, among 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, holdsAt and holdsFor predicates). Table I 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.

Table I. 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
holdsFor($F = V, Intervals$)	The value of fluent F is V during $Intervals$
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

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

$$\begin{aligned} &\mathsf{holdsAt}(\ F=V,\ T\ ) \leftarrow \\ &\quad \mathsf{initially}(\ F=V\ ), \\ &\quad \mathsf{not\ broken}(\ F=V,\ 0,\ T\ ) \end{aligned} \tag{1}$$

$$\begin{aligned} &\mathsf{holdsAt}(\ F=V,\ T\ ) \leftarrow \\ &\quad \mathsf{happens}(\ Act,\ T'\ ), \\ &\quad T' < T, \\ &\quad \mathsf{initiates}(\ Act,\ F=V,\ T'\ ), \\ &\quad \mathsf{not\ broken}(\ F=V,\ T',\ T\ ) \end{aligned} \tag{2}$$

According to axiom (1) 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 (2) 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 [10]. The domain-independent predicate broken is defined as follows:

$$\begin{aligned} &\mathsf{broken}(\ F=V,\ T_1,\ T_3\ ) \leftarrow \\ &\quad \mathsf{happens}(\ Act,\ T_2\ ), \\ &\quad T_1 \leq T_2,\ \ T_2 < T_3, \\ &\quad \mathsf{terminates}(\ Act,\ F=V,\ T_2\ ) \end{aligned} \tag{3}$$

$F = V$ 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:

$$\begin{aligned} &\mathsf{terminates}(\ Act,\ F=V,\ T\ ) \leftarrow \\ &\quad \mathsf{initiates}(\ Act,\ F=V',\ T\ ), \\ &\quad V \neq V' \end{aligned} \tag{4}$$

Axiom (4) 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 (1)–(4), specify that a fluent does not hold at the time it was initiated but does hold at the time it was terminated. Second, in addition to axioms (1)–(4), the domain-independent axioms of EC include those defining the holdsFor predicate, that is, the predicate for computing the intervals in which a fluent holds. To save space we do not present here the definition of holdsFor.

In the following sections we present a logic programming implementation of an Event Calculus action description, called EClp, expressing the NetBill specification. The complete source code of EClp is available upon request.

4. Social Constraints

We specify social constraints at design-time; furthermore, we assume that their specification does not change at run-time. There are several advantages to specifying social constraints at design-time [50]:

- Since the system designers' design-time resources are usually greater than the agents' run-time resources, some problems may be better solved by the designers at design-time than they would be solved at run-time by the agents.
- In case 'effective' social constraints are difficult to specify, the environment may be modified (at design-time) in a way that will simplify the task of devising social constraints (for example, adding traffic lights in a two-dimensional grid where mobile robots move).
- A design-time specification of social constraints may keep the agents from arriving at many of the conflicts to begin with. If the agents are aware of the social constraints before the commencement of the execution of an open MAS, they may produce their strategies according to the constraints and, therefore, avoid run-time conflicts.

Moreover, the existence of social constraints before the commencement of agent activities facilitates agents to decide whether or not to enter an open MAS. Similarly, agent designers may decide whether or not to deploy their agents in an open MAS based on the specification of social constraints. A framework for 'dynamic specifications', that is, specifications developed at design-time but modifiable at run-time, is an area of current work and is discussed in the last section of the paper.

We view open MAS from an external perspective (also referred to as *meta-perspective* by Werner [80, p.7]). Our specification is based only on externally observable states of affairs and not on the internals of the member agents. Furthermore, the specification of social constraints refers to the externally observable behaviour of the agents and not to the way agents reason about their behaviour. Apart from these externally specified constraints, agents (usually) have to comply with a different set of constraints, that is, a set of *internal constraints* imposed on them by their designers (say). These are part of the agents' architectures and may be in conflict with the social (external) constraints. In the NetBill protocol, for instance, a merchant may be permitted to present a quote that is associated with any possible price. Such a permission is externally specified, as part of the NetBill protocol. According to its own strategy, however, as determined by its designers, a merchant agent may not be 'permitted' to present a quote with a price that is less than the cost of the associated goods.

Note that agents may not be aware of all externally specified constraints of a MAS. The issue of making available the constraints of a MAS to the members is discussed in Section 6.

We maintain the standard, in the study of social and legal systems, long established distinction between permission, physical capability and institutional power. Jones and Sergot [38] cite Makinson [43] to illustrate the distinction between these concepts:

> "[C]onsider the case of a priest of a certain religion who does not have permission, according to instructions issued by the ecclesiastical authorities, to marry two people, only one of whom is of that religion, unless they both promise to bring up the children in that religion. He may nevertheless have the [*institutional*] *power* to marry the couple even in the absence of such a promise, in the sense that if he goes ahead and performs the ceremony, it still counts as a valid act of marriage under the rules of the same church even though the priest may be subject to reprimand or more severe penalty for having performed it." [43, p.409]

Jones and Sergot further point out that "one may imagine circumstances in which it is not *practically possible* for the priest to marry the couple (because, say, he is sick or otherwise incapacitated), although he is still empowered to do so" [38, p.431].

Accordingly, we present a four-level specification of the social constraints of an open MAS, that expresses:

- the physical capabilities (what is practically possible),
- institutional powers,
- permissions, prohibitions and obligations of the agents,
- the *sanctions* and *enforcement strategies* that deal with the performance of forbidden actions and non-compliance with obligations.

These levels of specification are presented in the following sections.

Recall that we express MAS specifications by means of EC action descriptions. The main actions and fluents of the EC action description expressing the NetBill specification are displayed in Tables II and III respectively. Note that we have simplified slightly. In general it could be that consumer C has several concurrent requests/contracts pending at any time to purchase goods GD at price P from merchant M, and we should add another parameter (a transaction ID) to each action and fluent in order to identify uniquely which specific request/purchase is being referred to in acceptance/payment actions. For simplicity we will ignore this minor complication. It is easily fixed (by adding a transaction ID), and is not important for the purposes of this paper.

4.1. Physical Capability

In this level of specification we express the agents' physical capabilities, that is, when is it practically possible for an agent to perform an action. Consider the following example.

Example 1 (Physical Capability) A consumer can send an electronic payment order (EPO) if it has the available funds:

$$\begin{aligned} &\mathsf{holdsAt}(\ can(C, send\_EPO(C, IS, GD, P)) = \mathsf{true},\ T\ ) \leftarrow \\ &\quad \mathsf{holdsAt}(\ bank\_account(C) = (Bal, Av),\ T\ ), \\ &\quad Av \geq P \end{aligned} \tag{5}$$

Physical capability is expressed with the *can* fluent (see Table III). $bank\_account(C) = (Bal, Av)$ records consumer C's current balance Bal (the total amount currently in the account) and available balance Av (the current balance plus overdraft protection funds). Clearly such information

Table II. Main Actions of the NetBill Specification.

Action	Textual Description
$request\_quote(C, M, GD)$	consumer C requests a quote from merchant M concerning goods described as GD
$present\_quote(M, C, GD, P)$	merchant M presents a quote to consumer C concerning goods described as GD at price P
$accept\_quote(C, M, GD, P)$	consumer C accepts a quote from merchant M concerning goods described as GD at price P
$send\_EPO(C, IS, GD, P)$	consumer C sends to the Intermediation Server IS an electronic payment order of amount P concerning the contract on goods described as GD
$send\_goods(M, IS, GD, G)$	merchant M sends to the Intermediation Server IS the goods G concerning the contract on GD

need not be available to the other NetBill participants. Axiom (5) expresses that an agent C can send an EPO of amount P if its available balance Av is greater or equal to P.

The representation of physical capability in our EC specification additionally includes axioms of the following form:

$$\begin{aligned} &incons(\ physical\_capability(send\_EPO(C, IS, GD, P))\ ) \leftarrow \\ &\quad \mathsf{happens}(\ send\_EPO(C, IS, GD, P),\ T\ ), \\ &\quad \mathsf{holdsAt}(\ can(C, send\_EPO(C, IS, GD, P)) = \mathsf{false},\ T\ ) \end{aligned} \tag{6}$$

incons is a predicate expressing a type of 'narrative inconsistency': a narrative of events is inconsistent if it includes actions that are not physically possible (at a particular time).

An alternative formalisation of the physical capability to send an EPO could have been the following: a consumer C is always capable of signalling $send\_EPO$, but the Intermediation Server is capable of transferring the amount P only when C's available balance is greater or equal to P.

Note that the specification of physical capability, and of actions more generally, abstracts away some detail. It may be that C does not in fact have the practical possibility of sending an EPO even when it has the available funds. C may have 'crashed', for instance, or be prevented for one reason or another from performing the physical act by means of which $send\_EPO$ is communicated (the communication channel may be blocked, for instance). Such information is not necessarily available (it may not be observable whether or not an agent has crashed) and thus is not considered by our specification of physical capability at the level of detail being modelled here. □

For this NetBill example we impose no further constraints on the capability to perform the remaining protocol actions.

In order to express the effects of a protocol action we need to distinguish between the act of bringing about a state of affairs (such as making an offer) and the act by means of which that state of affairs is brought about (such as sending a message with a particular form of words). Accepting a quote, by means of (say) sending a TCP/IP message, does not necessarily create a 'contractual obligation' (in a sense that will be made clear later) on the recipient to deliver the goods. It is only if the acceptance of the quote is signalled by an agent with the institutional

Table III. Main Fluents of the NetBill Specification.

Boolean Fluent	Textual Description
$can(Ag, Act)$	agent Ag is capable of performing Act
$pow(Ag, Act)$	agent Ag is empowered to perform Act
$per(Ag, Act)$	agent Ag is permitted to perform Act
$obl(Ag, Act)$	agent Ag is obliged to perform Act
$request(M, C, GD)$	consumer C requested a quote from merchant M regarding goods described as GD
$contract(M, C, GD, P)$	a contract has been established between merchant M and consumer C regarding goods described as GD at price P
$suspended(Role, Ag)$	agent Ag is suspended from acting as a $Role$
$violation(Role, Ag)$	agent Ag acting in $Role$ violated a particular social constraint

Non-Boolean Fluent	Domain	Textual Description
$bank\_account(Ag)$	$\mathbb{Z} \times \mathbb{N}$	agent Ag's current and available balance
$role\_of(Ag)$	$\mathcal{P}(\{consumer, merchant, iServer\})$	the roles that agent Ag occupies
$quote(M, C, GD, P)$	$\mathbb{N}$	the expiry time of merchant M's valid quote to consumer C regarding goods described as GD at price P
$ad\_count(M, C)$	$\mathbb{Z}$	the number of unsolicited quotes presented to consumer C by merchant M
$suspendedUntil(Role, Ag)$	$\mathbb{N}$	agent Ag is suspended from acting as a $Role$ until some future time

power to accept the quote that the recipient will be obliged to deliver the goods. The same act performed by an agent without this power has no effect on the recipient's obligations, though it may have other effects. An account of institutional power is presented next.

4.2. Institutional Power and Valid Actions

The term institutional (or 'institutionalised') power refers to the characteristic feature of an institution, legal system, formal organisation, or informal grouping, whereby designated agents, often when acting in specific roles, are empowered to create or modify facts of special significance in that institution, *institutional facts* [62,61], usually by performing a specified kind of act (such as when a priest performs a marriage, or an agent signs a contract, or the chairperson of a formal meeting declares the meeting closed).

According to the account given by Jones and Sergot [38], 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 raising a hand counts as making a bid at an auction).

In some circumstances it is unnecessary to isolate and name all instances of the acts by means of which agents exercise their institutional powers. In the NetBill protocol, for example, it is convenient to say that 'the consumer c accepted a quote q (issued by merchant m)' and let the context disambiguate whether we mean by this that the consumer performed an action, such as sending a message of a particular form via a TCP/IP socket connection, by means of which the acceptance of the quote is signalled, or whether the consumer actually issued an acceptance, in the sense that a contract is established between the consumer and the merchant. 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* to perform it at that point in time.

When we say that 'the consumer c accepted a quote q (issued by merchant m)' we mean, by convention, merely that c signalled its intention to accept q; this act was not necessarily effective in establishing a contract between c and m. In order to say that a contract has been established, we say that the action 'consumer c accepted a quote q' was *valid*: not only did c signal its intention to accept q, but also c was empowered to accept q at that time. Similarly, *invalid* is used to indicate lack of institutional power: when we say that the action 'consumer c accepted a quote q' is invalid we mean that c signalled its intention to accept q but did not have the institutional power to accept that quote at that time (and so the attempt to establish a contract was not successful).

We are aware that this use of the term 'valid' is not ideal and may be inappropriate in some contexts. Terms such as 'valid', 'in order', 'proper' (and 'invalid', 'out of order', 'improper') have specific meanings in particular contexts. However, these contexts are relatively few, and the same meaning is not always given in each. It is difficult to find a suitably neutral term; in this paper we will keep the term 'valid'.

The specification of institutional power is application-specific. Here, we present a few examples for the sake of a concrete NetBill illustration.

Example 2 (Institutional Power) A merchant is empowered to present a quote to any consumer about any type of good at any price:

$$\begin{aligned} &\mathsf{holdsAt}(\ pow(M, present\_quote(M, C, GD, P)) = \mathsf{true},\ T\ ) \leftarrow \\ &\quad \mathsf{holdsAt}(\ role\_of(M) = M\_Roles,\ T\ ),\quad merchant \in M\_Roles, \\ &\quad \mathsf{holdsAt}(\ role\_of(C) = C\_Roles,\ T\ ),\quad consumer \in C\_Roles \end{aligned} \tag{7}$$

According to the above axiom, an agent M is empowered to present a quote to an agent C about goods described as GD at price P if M occupies the role of merchant and C occupies the role of consumer. The Boolean fluent *pow* expresses the institutional power to perform an action. (Recall that Tables III and II present the intended informal readings of the fluents and actions, respectively, employed in the example EC specification.) The *role_of* fluent, whose value is a set of role names, records the set of roles each agent occupies at any time. The ways in which agents may acquire or lose roles are presented in later sections. □

Alternatively, we may choose to employ a stricter specification of the power to present a quote, as in the following example.

Example 3 (Institutional Power) A merchant is empowered to present a quote to a consumer about a type of good, provided that the consumer has requested a quote about that type

of good:

$$\begin{array}{l}\mathsf{holdsAt}(\ pow(M, present\_quote(M,C,GD,P)) = \mathsf{true},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ role\_of(M) = M\_Roles,\ T\ ),\quad merchant \in M\_Roles, \\ \quad \mathsf{holdsAt}(\ role\_of(C) = C\_Roles,\ T\ ),\quad consumer \in C\_Roles, \\ \quad \mathsf{holdsAt}(\ request(C,M,GD) = \mathsf{true},\ T\ )\end{array} \tag{8}$$

The *request* fluent records the pending requests of each consumer at any given time.

Suppose that C no longer occupies the consumer role after requesting a quote from merchant M. Then M is no longer empowered to present a quote to C since, according to the conditions (line 3) of axiom (8), a merchant is empowered to present quotes to consumers only. Similarly, if M ceases to occupy the merchant role then it loses all powers associated with that role, including the power to present quotes. □

Example 4 (Institutional Power) A consumer is empowered to accept a quote if that quote was issued in a valid manner by a merchant, and the quote has not expired.

$$\begin{array}{l}\mathsf{initiates}(\ present\_quote(M,C,GD,P),\ quote(M,C,GD,P) = T{+}10,\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ pow(M, present\_quote(M,C,GD,P)) = \mathsf{true},\ T\ )\end{array} \tag{9}$$

The fluent *quote* represents that a quote (with the specified parameters) is pending. It is initiated by a valid *present_quote* action, that is, by a *present_quote* performed by a merchant who currently has the power to issue such quotes. The value of a *quote* fluent records the time that the quote expires (say 10 time-points after the issue of the quote).

The power to accept a quote is expressed as follows:

$$\begin{array}{l}\mathsf{holdsAt}(\ pow(C, accept\_quote(C,M,GD,P)) = \mathsf{true},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ role\_of(C) = C\_Roles,\ T\ ),\quad consumer \in C\_Roles, \\ \quad \mathsf{holdsAt}(\ role\_of(M) = M\_Roles,\ T\ ),\quad merchant \in M\_Roles, \\ \quad \mathsf{holdsAt}(\ quote(M,C,GD,P) = QuoteT,\ T\ ), \\ \quad QuoteT \geq T\end{array} \tag{10}$$

The last two conditions of the above axiom (lines 4–5) require that a quote has not expired. Note, again, that if M ceases to occupy the merchant role after issuing a valid quote then C loses its power to accept M's quote, even if the quote has not expired. Furthermore, C will be empowered to accept quotes as long as it occupies the consumer role. □

Accepting a quote when having the power to do so, establishes a contract between the two parties in question:

$$\begin{array}{l}\mathsf{initiates}(\ accept\_quote(C,M,GD,P),\ contract(M,C,GD,P) = \mathsf{true},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ pow(C, accept\_quote(C,M,GD,P)) = \mathsf{true},\ T\ )\end{array} \tag{11}$$

A contract creates a set of further powers of the contracting parties; these are expressed similarly to the ones presented above. Additionally, a contract creates a bundle of permissions and obligations, one of which is presented in the following section (Example 8).

4.3. Permission and Obligation

We now specify what actions, valid or invalid, are to be classified as permitted or obligatory. Such a specification is application-specific. In some cases, permissions are associated with institutional powers.

Example 5 (Permission) An agent is permitted to perform an action if it is empowered to perform that action:

$$\begin{aligned}\mathsf{holdsAt}(\ per(Ag, Act) = \mathsf{true},\ T\ ) \leftarrow \\ \mathsf{holdsAt}(\ pow(Ag, Act) = \mathsf{true},\ T\ )\end{aligned} \tag{12}$$

The fluent *per* expresses permission. □

In the example above we have chosen to say that an agent is always permitted to exercise its institutional powers. In other examples the relationship between power and permission could be even stronger: we might prefer to say that an agent is permitted to perform an action if *and only if* it is empowered to perform that action. In general there is no standard, fixed relationship between power and permission. For instance, it is sometimes valuable to permit an agent Ag to perform an action Act even if Ag is not empowered to perform Act, or to forbid Ag from performing Act even if Ag is empowered to perform Act. Consider the following examples.

Example 6 (Permission) A merchant is permitted to present a quote to a consumer if the consumer has requested a quote, or the merchant has not already issued 'too many' unsolicited quotes.

$$\begin{aligned}&\mathsf{holdsAt}(\ per(M, present\_quote(M, C, GD, P)) = \mathsf{true},\ T\ ) \leftarrow \\ &\quad \mathsf{holdsAt}(\ role\_of(M) = M\_Roles,\ T\ ),\quad merchant \in M\_Roles, \\ &\quad \mathsf{holdsAt}(\ role\_of(C) = C\_Roles,\ T\ ),\quad consumer \in C\_Roles, \\ &\quad \mathsf{holdsAt}(\ request(C, M, GD) = \mathsf{true},\ T\ )\end{aligned} \tag{13}$$

$$\begin{aligned}&\mathsf{holdsAt}(\ per(M, present\_quote(M, C, GD, P)) = \mathsf{true},\ T\ ) \leftarrow \\ &\quad \mathsf{holdsAt}(\ role\_of(M) = M\_Roles,\ T\ ),\quad merchant \in M\_Roles, \\ &\quad \mathsf{holdsAt}(\ role\_of(C) = C\_Roles,\ T\ ),\quad consumer \in C\_Roles, \\ &\quad \mathsf{holdsAt}(\ ad\_count(M, C) = Count,\ T\ ), \\ &\quad Count < 10\end{aligned} \tag{14}$$

Axiom (13) states that it is permitted to present a quote to a consumer that has requested one. (Recall that the *request* fluent records pending consumer requests.) Axiom (14) states that a merchant M is permitted to present a quote, provided that M has not already issued 'too many' unsolicited quotes. The $ad\_count(M, C)$ fluent records the number of unsolicited quotes merchant M has currently issued to consumer C. It is incremented whenever an unsolicited quote is presented, expressed as follows:

$$\begin{aligned}&\mathsf{initiates}(\ present\_quote(M, C, GD, P),\ ad\_count(M, C) = AC{+}1,\ T\ ) \leftarrow \\ &\quad \mathsf{holdsAt}(\ request(C, M, GD) = \mathsf{false},\ T\ ) \\ &\quad \mathsf{holdsAt}(\ ad\_count(M, C) = AC,\ T\ )\end{aligned} \tag{15}$$

The value might also be decremented, say after a specified time has elapsed. The details will be application-specific. (The full source of this example specification, available on request, includes rules for determining $ad\_count(M, C)$ for the sake of a concrete illustration.)

This type of permission may be specified in order to protect consumers from the merchants' over-advertising their goods.

Suppose that the power to present a quote is expressed as in Example 2. In this case, a merchant M may be empowered to present a quote to a consumer C but forbidden to exercise this power — if, for example, C has not requested a quote and M has issued 'too many' unsolicited quotes in the past. Exercising this power will empower C to accept the quote (see Example 4) but

M may be subject to penalty for performing this forbidden action (sanctions and enforcement strategies are discussed in the following section).

If on the other hand the power to present a quote is expressed as in Example 3 then M may be permitted but not empowered to present a quote. For instance, M is permitted to issue a quote to C although C has not requested one, provided that $ad\_count(M, C) < 10$. In this case presenting a quote will not empower C to accept it, but M will not be subject to penalty for advertising its goods. □

Example 7 (Permission) A consumer is permitted to accept a quote if it is empowered to do so, *and* it can afford the associated price.

$$\begin{array}{l} \mathsf{holdsAt}(\ per(C, accept\_quote(C, M, GD, P)) = \mathsf{true},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ pow(C, accept\_quote(C, M, GD, P)) = \mathsf{true},\ T\ ), \\ \quad \mathsf{holdsAt}(\ bank\_account(C) = (Bal, Av),\ T\ ), \\ \quad Av \geq P \end{array} \tag{16}$$

Recall that the *bank_account* fluent expresses an agent's current and available balance. This information may be seen as a certificate that a consumer C must provide to the entity computing permissions, powers, etc, so that it can be determined whether C is permitted or not to accept a quote. (A discussion about the run-time computation of a protocol state, including the agents' permissions current at each time, is presented in Section 6.) As already mentioned, the details of an agent's bank account need not be available to the other NetBill participants.

According to this example, a consumer may be empowered to accept a quote (see Example 4) but forbidden to exercise this power if it cannot afford the associated price. Exercising this power *will* create a contract (see axiom (11)) but the consumer may be subject to penalty for violating this prohibition.

This type of permission could be specified in order to deter agents from establishing a contract when there is a high probability that they will not be able to comply with its terms, in this case to pay the merchant when the goods are delivered. A stricter specification would require that a consumer's current balance, rather than its available balance, should be greater or equal to the quoted price (that is, the last line of axiom (16) being $Bal \geq P$). □

Note that, like institutional power, permission does not imply (and is not implied by) physical capability. It could be that a consumer, though permitted to accept a quote, may nevertheless be unable to accept that quote, because it is (temporarily perhaps) unable to perform the physical act by means of which the acceptance is communicated.

Other example specifications of permission in NetBill may include that a merchant is permitted to sell a type of good up to a specified price (for example, songs in 128kbps mp3 format should not be priced more than €0.99), a merchant is forbidden to advertise the same type of good at different prices to different consumers, and so on. A contract creates a set of further permissions/prohibitions to the contracting parties; for example, the consumer is forbidden to send a payment order that does not cover the agreed amount, the merchant is forbidden to deliver goods that do not meet the agreed quality, and so on.

In the NetBill example, a contract creates a set of obligations on the contracting parties:

Example 8 (Obligation) A consumer is obliged to pay the agreed price to the contracting merchant by a specified deadline.

Accepting a quote, while having the power to do so (that is, establishing a contract), obliges the consumer to pay the agreed price:

$$\begin{array}{l}\mathsf{initiates}(\ accept\_quote(C,M,GD,P),\ obl(C,send\_EPO(C,IS,GD,P)) = \mathsf{true},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ pow(C,accept\_quote(C,M,GD,P)) = \mathsf{true},\ T\ ), \\ \quad \mathsf{holdsAt}(\ role\_of(IS) = IS\_Roles,\ T\ ),\quad iServer \in IS\_Roles\end{array} \tag{17}$$

The *obl* fluent expresses obligation. The above obligation is discharged by sending an EPO of (at least) the agreed amount:

$$\begin{array}{l}\mathsf{initiates}(\ send\_EPO(C,IS,GD,P),\ obl(C,send\_EPO(C,IS,GD,P')) = \mathsf{false},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ obl(C,send\_EPO(C,IS,GD,P')) = \mathsf{true},\ T\ ), \\ \quad P \geq P'\end{array} \tag{18}$$

The obligation should be discharged by a specified deadline. This is represented in our specification by a *timeout event* that takes place at a specified time (the time by which the obligation should be discharged) after the valid acceptance of a quote. The timeout, which has as parameter the goods description of the accepted quote, terminates the aforementioned obligation, if the obligation exists at the time of the timeout:

$$\begin{array}{l}\mathsf{initiates}(\ timeout(GD),\ obl(C,send\_EPO(C,IS,GD,P)) = \mathsf{false},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ obl(C,send\_EPO(C,IS,GD,P)) = \mathsf{true},\ T\ )\end{array} \tag{19}$$

If the obligation does hold at the time of the timeout, that is, if the obligation has not been discharged, then the timeout will have additional effects: the application of sanctions to the consumer (see the following section). □

In a similar way we may specify the merchant's obligation to deliver the goods.

In general, like powers and permissions, the specification of obligations is application-specific. It is important, however, to maintain consistency of the specification of permissions and obligations on the same MAS. For instance, an agent should not be forbidden and obliged to perform the same action at the same time. In the NetBill example, a consumer should not be obliged to send an EPO of amount P and permitted to send an EPO of amount less than P, and so on.

Determining what actions are permitted, prohibited or obligatory enables the classification of the behaviour of individual agents and the MAS as a whole into categories such as 'social' or 'anti-social', 'acceptable' or 'unacceptable', 'desirable' or 'undesirable', and so on. Usually, the behaviour of an agent is considered 'anti-social' or 'unacceptable' if that agent performs (certain) forbidden actions or does not comply with (some of) its obligations. Furthermore, based on the behaviour of the individual agents, it is possible to classify the behaviour of the MAS. For instance, the state of a MAS may be considered 'unacceptable' at a point in time if the majority of its members have violated their obligations at that point in time.

4.4. Sanction

The fourth level of specification expresses the sanctions and enforcement strategies that deal with 'anti-social' or 'undesirable' behaviour. We are concerned with the following issues:

1. when is an agent sanctioned, and

2. what is the penalty that an agent has to face (in the case that it does get sanctioned).

The specification of both of these issues is application-dependent.

In the presented NetBill example we want to reduce or eliminate, and therefore penalise, the following types of 'undesirable' behaviour:

- where the merchant presents a quote, or the consumer accepts a quote, when forbidden to do so, and
- non-compliance with a contract's obligations (the consumer fails to pay the merchant the agreed price, or the merchant fails to deliver goods exactly as described in the contract).

The penalties for such 'sanctionable' behaviour may come in many different forms. The following examples illustrate some of the possibilities. We consider the aforementioned types of behaviour in turn.

Example 9 (Sanction) A merchant presents a quote when forbidden to do so.

The following axiom expresses a possible penalty for this type of 'sanctionable' behaviour:

$$\begin{array}{l}\mathsf{initiates}(\ present\_quote(M,C,GD,P),\ bank\_account(M) = (Bal - f, Av - f),\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ per(M, present\_quote(M,C,GD,P)) = \mathsf{false},\ T\ ), \\ \quad \mathsf{holdsAt}(\ bank\_account(M) = (Bal, Av),\ T\ ), \\ \quad Bal \geq f\end{array} \tag{20}$$

Here a fine f is deducted from the merchant M's bank account whenever M presents a quote while being forbidden to do so. If the fine f is greater than the merchant M's current balance then the amount deducted from M's account is $f+f\times i$, that is, an interest i on f is also deducted:

$$\begin{array}{l}\mathsf{initiates}(\ present\_quote(M,C,GD,P), \\ \quad bank\_account(M) = (Bal - (f+f\times i), Av - (f+f\times i)),\ T\ ) \leftarrow \\ \quad\quad \mathsf{holdsAt}(\ per(M, present\_quote(M,C,GD,P)) = \mathsf{false},\ T\ ), \\ \quad\quad \mathsf{holdsAt}(\ bank\_account(M) = (Bal, Av),\ T\ ), \\ \quad\quad Bal < f, \quad Av \geq (f+f\times i)\end{array} \tag{21}$$

Notice that axioms (20) and (21) do not cover the case where the merchant has insufficient funds to pay the fine, that is, the case where $Av < f$. We will comment on that presently. First we want to point out that axioms (20) and (21) are just an approximation of what we want to specify. They do not refer to the actions of paying a fine or transferring funds. They say merely that a merchant's bank balance decreases by a certain amount in the specified circumstances. They do not say how the balance is decreased, how the funds are transferred, or to whom. This is because we have simplified the example in order to keep it manageable. In a more complete formalisation the Intermediation Server (IS) and the banks shown in Figure 1 would also be included as roles in the NetBill protocol, and we would specify the possible actions (practical capabilities), powers, permissions, and obligations for agents occupying those roles in similar style to those shown for the consumer and merchant. We omit these details since they require more space than we have available here. There are many possibilities depending on how bank accounts are administered and managed. For example, one possible specification for NetBill is that (i) IS is empowered (in specified circumstances) to request (or demand, or simply effect) the transfer of funds from the merchant's and consumer's bank accounts, and (ii) IS has an obligation to exercise this power in certain circumstances (such as when a merchant is required to pay a fine). The specification can be formalised in exactly the same manner as that shown for the merchant and consumer roles. For the simplified version of the example presented in this paper, we assume that the IS always fulfills its obligations, and we represent the effects of its actions on the merchant's bank account without referring to its actions explicitly.

Consider now the remaining case $Av < f$ in which merchant M does not have sufficient funds available to pay the fine f. According to axioms (20) and (21), M could present quotes when forbidden to do so without facing any penalty. There are several ways of dealing with this case. Recall that a merchant is forbidden to issue 'too many' unsolicited quotes (in the example, more than 10 to any one customer). Suppose, for the sake of a concrete example, that if the number of unsolicited quotes made by a merchant M to a customer C exceeds a specified limit (say when $ad\_count(M, C) \geq 100$) then the merchant M becomes *suspended*, that is, loses its power to present a quote. The conditions for this form of suspension can be expressed as follows:

$$\begin{array}{l} \mathsf{holdsAt}(\ suspended(merchant, M) = \mathsf{true},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ role\_of(M) = M\_Roles,\ T\ ),\quad merchant \in M\_Roles, \\ \quad \mathsf{holdsAt}(\ ad\_count(M, C) = Count,\ T\ ), \\ \quad Count \geq 100 \end{array} \tag{22}$$

$suspended(merchant, M) = \mathsf{true}$ expresses that agent M is a suspended merchant. (Suspended consumers can be treated likewise.)

The effects of suspension on the merchant's power to present a quote are expressed (assuming the specification in Example 2) by adjusting axiom (7) as follows:

$$\begin{array}{l} \mathsf{holdsAt}(\ pow(M, present\_quote(M, C, GD, P)) = \mathsf{true},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ role\_of(M) = M\_Roles,\ T\ ),\quad merchant \in M\_Roles, \\ \quad \mathsf{holdsAt}(\ role\_of(C) = C\_Roles,\ T\ ),\quad consumer \in C\_Roles, \\ \quad \mathsf{holdsAt}(\ suspended(merchant, M) = \mathsf{false},\ T\ ) \end{array} \tag{7'}$$

The alternative specification of power to present a quote expressed by axiom (8) in Example 3 can be adjusted similarly.

A suspended merchant, that is, a merchant without the institutional power to present quotes, has no means of empowering consumers to accept its quotes and thus no means of establishing contracts. The penalty, however, is not permanent. The value of the $ad\_count(M, C)$ fluent may decrease over time. M would thereby cease to be suspended and would re-acquire its powers.

Naturally becoming a suspended merchant has implications on the powers and permissions that are due to the agent's role as a merchant; the powers and permissions that an agent may have due to other roles it occupies are not affected. □

Example 10 (Sanction) A consumer accepts a quote when forbidden to do so.

We record violations of this prohibition as follows:

$$\begin{array}{l} \mathsf{initiates}(\ accept\_quote(C, M, GD, P),\ violation(consumer, C) = \mathsf{true},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ pow(C, accept\_quote(C, M, GD, P)) = \mathsf{true},\ T\ ), \\ \quad \mathsf{holdsAt}(\ per(C, accept\_quote(C, M, GD, P)) = \mathsf{false},\ T\ ) \end{array} \tag{23}$$

(In this example we are not concerned with the case in which a consumer is forbidden but not empowered to accept a quote.) The fluent $violation(R, Ag)$ records Ag's violation of a social constraint when acting in role R. Exercising the power to accept a quote while being forbidden to do so is not necessarily penalised. In particular, for this example specification we chose to say that if the consumer C pays the contracting merchant M the agreed price then C's violation of the prohibition will be ignored. On the other hand, if C fails to pay the agreed price then it will face a more severe penalty (for failing to pay the price) than it would have had it not violated the aforementioned prohibition. The following example presents this type of penalty. □

Example 11 (Sanction) A consumer does not comply with its obligation to pay the merchant.

We could specify various penalties for the failure to pay the agreed price by the specified deadline. For instance, we could apply a monetary penalty when the consumer violates its obligation to pay, that is, a fine could be deducted from the consumer's account whenever this occurs. The formalisation of such a penalty would be similar to the one expressed by axioms (20) and (21) in Example 9. Alternatively (or additionally), depending on whether or not the consumer C had earlier violated the prohibition to accept a quote, C could be suspended, disqualified or even banned from NetBill for failing to comply with the obligation to pay the agreed price. Consider the following formalisation:

$$\begin{array}{l}\mathsf{initiates}(\ timeout(GD),\ suspendedUntil(consumer, C) = T{+}10,\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ obl(C, send\_EPO(C, IS, GD, P)) = \mathsf{true},\ T\ ), \\ \quad \mathsf{holdsAt}(\ violation(consumer, C) = \mathsf{false},\ T\ )\end{array} \tag{24}$$

$suspendedUntil(consumer, C) = T'$ expresses that C is a suspended consumer until time T'. The above axiom states that C's violation of its obligation to send an Electronic Payment Order (EPO) at T suspends C until $T{+}10$ (say). A suspended consumer loses temporarily the power to accept a quote (and thus establish a contract):

$$\begin{array}{l}\mathsf{holdsAt}(\ pow(C, accept\_quote(C, M, GD, P)) = \mathsf{true},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ role\_of(C) = C\_Roles,\ T\ ),\quad consumer \in C\_Roles, \\ \quad \mathsf{holdsAt}(\ role\_of(M) = M\_Roles,\ T\ ),\quad merchant \in M\_Roles, \\ \quad \mathsf{holdsAt}(\ quote(M, C, GD, P) = QuoteT,\ T\ ), \\ \quad QuoteT \geq T, \\ \quad \mathsf{holdsAt}(\ suspendedUntil(consumer, Consumer) = SusT,\ T\ ), \\ \quad SusT < T\end{array} \tag{10$'$}$$

Axiom (10$'$) is a modification of axiom (10), expressing the power to accept a quote, capturing the possibility that a consumer may be suspended.

The penalty expressed by axioms (24) and (10$'$) concerns the case in which the agent occupying the consumer role violated the obligation to send an EPO but had not earlier violated the prohibition to accept the quote (see the last condition of axiom (24)). If the agent had additionally violated this prohibition then it would have lost the consumer role:

$$\begin{array}{l}\mathsf{initiates}(\ timeout(GD),\ role\_of(C) = C\_Roles \setminus \{consumer\},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ obl(C, send\_EPO(C, IS, GD, P)) = \mathsf{true},\ T\ ), \\ \quad \mathsf{holdsAt}(\ violation(consumer, C) = \mathsf{true},\ T\ ), \\ \quad \mathsf{holdsAt}(\ role\_of(C) = C\_Roles,\ T\ )\end{array} \tag{25}$$

Losing the consumer role means that all powers, permissions and obligations associated with that role are lost. (This is in contrast to becoming a suspended consumer whereby one (temporarily) loses some or all powers and permissions associated with the consumer role, but retains the non-discharged obligations of that role.) A more severe penalty would have been the disqualification of the agent from NetBill, that is, the agent would lose *all* the roles it occupied. Disqualification may or may not be accompanied by a ban. A disqualified agent may re-apply to enter NetBill; a banned agent may not. This issue is discussed in Section 5.

Note that the violation of the obligation to send an EPO may or may not cancel the transaction between the contracting parties. (For instance, the merchant may be in favour of the completion of the transaction although the consumer has paid a smaller amount than the agreed one, in order to cash the consumer's payment.) The formalisation of this issue is not presented here. □

Similarly we may specify penalties for the remaining types of 'sanctionable' behaviour, including a merchant's non-compliance with its obligation to deliver the goods.

An additional or alternative type of penalty could be expressed in terms of *bad reputation* — for instance, a consumer not fulfilling its obligation to pay the merchant may receive 'negative feedback'. Reputation mechanisms are frequently used in open MAS (see [15, 83, 60] for a few examples). We will not discuss any examples of such mechanisms in this paper.

Sanctions are one means by which an open MAS may discourage 'undesirable' or 'anti-social' behaviour. Another possible enforcement strategy is to try to devise additional controls that will force agents to comply with their obligations or prevent them from performing forbidden actions. In the NetBill example, the forbidden presentation of a quote may be physically blocked. The general strategy of designing mechanisms to force compliance and eliminate non-permitted behaviour is what Jones and Sergot [37] referred to as *regimentation*. Regimentation devices have often been employed in order to eliminate 'undesirable' behaviour in computational systems. *Interagents* [56], for example, enforce the rules of the FishMarket auction house to the buyer and seller agents. *Sentinels* [39] monitor and, when necessary, modify some aspects of the agent interactions in order to provide 'exception handling' services. *Controllers* [48] enforce the 'law-governed interaction' regulatory mechanism in open MAS. It has been argued [37], however, that regimentation is rarely desirable (it results in a rigid system that may discourage agents from entering it [55]), and not always practical. 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 non-compliance and sanctioning and not rely exclusively on regimentation mechanisms.

5. Social Roles

A social role r is associated with a set of social constraints prescribing the behaviour of the agents occupying r. These constraints express the powers, permissions and obligations associated with r. Axioms (10) and (16)–(19), for instance, express a few powers, permissions and obligations associated with the consumer role.

A role r is also associated with a set of 'conditions' that an agent must satisfy in order to be *eligible* to occupy r. It should be possible to determine whether or not an agent satisfies the conditions of a role without having to access its internals. In NetBill, for example, we may specify that an agent satisfies the conditions of the consumer role if its current bank balance is over a specified limit.

To occupy a role in an open MAS, agents usually participate in a *role-management protocol*. Protocols of this form are typical in distributed systems — in resource sharing applications in the fields of Collaborative Multimedia Computing (CMC) and Computer-Supported Co-operative Work (CSCW), for example, *session control protocols* [18,19] prescribe, among other things, ways for joining, withdrawing from, inviting to join, and excluding from, a session (in which resources are shared/accessed). Accordingly, we present a simple role-management protocol that specifies the ways in which an agent may join, or withdraw from NetBill, and the conditions in which an agent is (temporarily or permanently) removed from NetBill. Table (IV) presents the actions and a number of fluents of the EC action description expressing the role-management protocol specification. (As will be shown later, there are a number of fluents in common between the EC action descriptions expressing NetBill and its role-management protocol.)

Table IV. Actions and Fluents of a Role Management Protocol in the Context of NetBill.

Action	**Textual Description**
$apply(Ag, RAA, Role)$	agent Ag applies for $Role$ to role assigning authority RAA
$assign(RAA, Ag, Role)$	role assigning authority RAA assigns $Role$ to agent Ag
$withdraw(Ag, RAA, Role)$	agent Ag submits its resignation from $Role$ to role assigning authority RAA

Fluent	**Domain**	**Textual Description**
$role\_cond(Role, Ag)$	Boolean	agent Ag satisfies the conditions of $Role$
$applied(Role, Ag)$	Boolean	agent Ag has a pending valid application for $Role$
$merchants$	$\mathbb{N}$	the number of agents occupying the merchant role in NetBill
$consumers$	$\mathbb{N}$	the number of agents occupying the consumer role in NetBill

According to a typical procedure for joining an open MAS, that is, acquiring a role in that MAS, agents apply for roles and a role assigning authority accepts or rejects applications. The decision-making of the role assigning authority is informed by, among other things:

– whether or not an applicant satisfies the conditions of a role,

– whether or not an applicant has been banned from a MAS (see Example 11), and

– whether or not the assignment of a role to an agent is consistent with the 'structure' of a MAS. In NetBill, for example, we may require (say) that the number of consumers is at most twice the number of merchants.

A role assigning authority may additionally consider the issue of 'role conflict' [36] or 'flow down' [70], that is, the case in which an agent occupies roles with conflicting permissions and obligations. One way to avoid a role conflict is to set the role conditions in such a way that it is impossible for an agent to satisfy at the same time the conditions of roles with conflicting permissions and obligations. Consequently, a role assigning authority would not proceed to role assignments that create role conflicts. This solution requires, however, that an agent satisfies the conditions of a role as long as it occupies the role (as opposed to satisfying the role conditions only when role assignment is considered).

Example procedures for deciding to award or deny a role are:

– chair-designated: an elected agent assigns roles,

– election: the members of a role assigning committee (comprised of existing MAS members or other elected agents) vote on role assignment,

– argumentation: the members of a role assigning committee debate on role assignment,

– lottery scheduling: role assignment operates on a probabilistic basis.

Formalisations of voting and argumentation procedures may be found in [53, 1] respectively. A simple example of role assignment, in the context of NetBill, is presented next.

Example 12 (Role Assignment) An agent satisfies the conditions of the consumer role if its current balance is greater or equal to 1000:

$$\begin{array}{l}\mathsf{holdsAt}(\ role\_cond(consumer, C) = \mathsf{true},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ bank\_account(C) = (Bal, Av),\ T\ ), \\ \quad Bal \geq 1000\end{array} \tag{26}$$

$role\_cond(R, Ag) = \mathsf{true}$ expresses that agent Ag satisfies the conditions of role R (see Table IV).

Only valid applications for roles are considered; the power to apply for the consumer role is expressed as follows:

$$\begin{array}{l}\mathsf{holdsAt}(\ pow(C, apply(C, RAA, consumer)) = \mathsf{true},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ role\_of(C) = C\_Roles,\ T\ ),\quad consumer \notin C\_Roles, \\ \quad \mathsf{holdsAt}(\ role\_of(RAA) = RAA\_Roles,\ T\ ),\quad authority \in RAA\_Roles, \\ \quad \mathsf{holdsAt}(\ applied(consumer, C) = \mathsf{false},\ T\ ), \\ \quad \mathsf{holdsAt}(\ role\_cond(consumer, C) = \mathsf{true},\ T\ ), \\ \quad \mathsf{holdsAt}(\ banned(C) = \mathsf{false},\ T\ )\end{array} \tag{27}$$

An agent C is empowered to apply for the consumer role to a role assigning authority RAA if C:

- does not already occupy that role (line 2 of the above axiom),
- does not have a pending application for that role (line 4) — the *applied* fluent records pending valid role applications,
- satisfies the conditions of that role (line 5), and
- has not been banned from NetBill — denoted by $banned(C) = \mathsf{false}$ (line 6).

We have specified the power of a role assigning authority RAA to assign the consumer role to an agent as follows:

$$\begin{array}{l}\mathsf{holdsAt}(\ pow(RAA, assign(RAA, C, consumer)) = \mathsf{true},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ role\_of(RAA) = RAA\_Roles,\ T),\quad authority \in RAA\_Roles, \\ \quad \mathsf{holdsAt}(\ applied(consumer, C) = \mathsf{true},\ T\ ), \\ \quad \mathsf{holdsAt}(\ merchants = M,\ T\ ), \\ \quad \mathsf{holdsAt}(\ consumers = C,\ T\ ), \\ \quad M \geq ((C{+}1)/2)\end{array} \tag{28}$$

According to the above axiom, the conditions for being empowered to assign the consumer role to an agent C include that C has a pending application for that role, and a role assignment is in accordance with the 'structure' of the MAS; in this example, the number of consumers is at most twice the number of merchants. The fluents *merchants* and *consumers* indicate the number of existing merchants and consumers respectively.

For simplicity, in this example we do not consider the issue of role conflict.

Assume that a role assigning authority is indeed empowered to assign the consumer role to an agent. The decision to proceed to, or refrain from, the assignment may be based on one of the aforementioned procedures (voting, argumentation, etc). The reader is referred to the cited papers for examples of formalisations of such procedures.

The direct effects of a valid role assignment are expressed in terms of the *role_of* fluent. In this example, the applicant occupies the consumer role:

$$\begin{array}{l}\mathsf{initiates}(\ assign(RAA, C, consumer),\ role\_of(C) = \{consumer\} \cup C\_Roles,\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ pow(RAA, assign(RAA, C, consumer)) = \mathsf{true},\ T\ ), \\ \quad \mathsf{holdsAt}(\ role\_of(C) = C\_Roles,\ T\ )\end{array} \tag{29}$$

A valid role assignment also has indirect effects on the *pow*, *per* and *obl* fluents: the successful applicant now holds the powers, permissions and obligations of a consumer. □

In addition to expressing the process of role assignment, a role-management protocol prescribes the ways in which an agent may successfully withdraw from a role, and describes the conditions in which an agent gets disqualified or banned from a MAS. Withdrawing from a role, or losing a role (due to, say, disqualification or ban), implies losing the powers, permissions and obligations of that role. An example procedure of role withdrawal is presented next.

Example 13 (Role Withdrawal) An agent is empowered to withdraw from the merchant role if it: (i) occupies that role, (ii) has no pending quotes, and (iii) has no pending obligations:

$$\begin{array}{l}\mathsf{holdsAt}(\ pow(M, withdraw(M, RAA, merchant)) = \mathsf{true},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ role\_of(M) = M\_Roles,\ T),\quad merchant \in M\_Roles, \\ \quad \mathsf{holdsAt}(\ role\_of(RAA) = RAA\_Roles,\ T),\quad authority \in RAA\_Roles, \\ \quad \mathsf{not}\ (\ \mathsf{holdsAt}(\ quote(M, C, GD, P) = QT,\ T\ ),\quad QT \geq T\ ), \\ \quad \mathsf{not}\ \mathsf{holdsAt}(\ obl(M, send\_goods(M, C', GD', P')) = \mathsf{true},\ T\ )\end{array} \tag{30}$$

The penultimate condition of the above axiom (line 4) requires that all quotes of the merchant agent (if any) have expired. If there is a quote that has not expired then an agent may not successfully withdraw from the merchant role because a consumer may accept the quote and thus create a contract between them. Clearly this condition to withdrawal can be lifted in other examples. According to the last condition of the above axiom (line 5), an agent that has pending obligations to deliver goods may not withdraw from the merchant role and thus avoid discharging these obligations. In general, an agent that has pending obligations due to a role that it occupies may not withdraw from that role as long as these obligations are not terminated (that is, discharged, or violated and sanction applied). □

Considering the issue of (temporarily or permanently) losing a role, in Section 4.4 we showed ways in which an agent may be disqualified or even banned from a MAS, due to its 'anti-social' behaviour. An additional reason for role revocation, or role suspension, could be the failure to satisfy the conditions of the role one occupies. In other words, it may be required, for a particular MAS, that an agent satisfies the conditions of a role as long as it occupies that role. A formalisation of role suspension due to failure to satisfy the role conditions is the following:

$$\begin{array}{l}\mathsf{holdsAt}(\ suspended(Role, Ag) = \mathsf{true},\ T\ ) \leftarrow \\ \quad \mathsf{holdsAt}(\ role\_of(Ag) = Ag\_Roles,\ T\ ),\quad Role \in Ag\_Roles, \\ \quad \mathsf{holdsAt}(\ role\_cond(Role, Ag) = \mathsf{false},\ T\ )\end{array} \tag{31}$$

For instance, a consumer will be suspended as long as its current balance is less than 1000 (see axiom (26) for the conditions of the consumer role). Similarly, we may express role revocation due to failure to satisfy the role conditions.

Note that 'anti-social' behaviour and the failure to satisfy the role conditions are not the only ways of losing a role. For example, a role-management protocol may allow for time-bound

memberships, that is, roles are assigned for a specified time period, after which the agent loses the associated powers, permissions and obligations.

In general, a specification of a more complex role-management protocol would express, among other things:

- additional procedures for role-management such as inviting to join a MAS;
- the conditions in which an agent is said to be permitted, or obliged, to apply for, assign, or withdraw from, a role (or perform any other action of a role-management protocol), and
- the sanctions that are applied in the case of non-conformance with the aforementioned permissions and obligations.

Such a specification would be similar to that of NetBill presented in the previous sections or to other protocol specifications presented elsewhere [2, 1]. We omit the details here to save space.

6. Executing the Specification

6.1. Protocol Animation

In order to illustrate our framework for executable specifications, in this section we present an example run of NetBill and the associated role-management protocol. A part of the narrative of events of this run is displayed in the first two columns of Table V; the messages concerning the role-management protocol are indented whereas the messages concerning NetBill are not. The third column of Table V shows the roles of each NetBill participant; for brevity the merchant role is denoted by m and the consumer role by c. Given a set of temporally-ordered events (a narrative), we may query our Event Calculus logic programming (EClp) implementation to determine the system state current at each time. For example, to find out whether ag_1 is empowered to present a quote at time-point 15, we compute the following query:

$$? - \mathsf{holdsAt}(\ pow(ag_1, present\_quote(ag_1, C, GD, P)) = \mathsf{true},\ 15\ )$$

We discuss next the system states of the run displayed in Table V.

Initially, agent ag_1 occupied the role of merchant, ag_2 occupied the role of consumer, and ag_3 occupied both roles (see the third column of Table V). At time-point 15 ag_1 presented a quote to ag_3 concerning $book_A$ at price *23*. According to axiom (7), which is part of this example NetBill specification, ag_1 was empowered to perform the aforementioned action because it occupied the role of merchant and ag_3 occupied the role of consumer. Consequently, according to axioms (9) and (10), ag_3 become empowered at time-point 16 to accept the quote of ag_1. Indeed, ag_3 accepted the quote of ag_1 at time-point 22, thus establishing a contract between them (see axiom (11)), creating a bundle of powers, permissions and obligations for them. At time-point 24 ag_1 sent the goods to the Intermediation Server (*iServer*) while at the next time-point ag_3 attempted to withdraw from the consume role: ag_3 sent a *withdraw* message to the role assigning authority *raa*. This attempt was not successful because ag_3 was not empowered to withdraw from the consumer role at the time, since ag_3 had a pending obligation to send an electronic payment order (EPO) concerning the contract on $book_A$ (see Example 13).

At time-point 28 ag_3 attempted to discharge its obligation by sending an EPO; the amount, however, was less than the agreed one (*13* instead of *23*) and thus the obligation was not

Table V. A Sample Run of NetBill and its Role-Management Protocol.

Time	Action	Roles
15	$present\_quote(ag_1, ag_3, book_A, 23)$	ag_1:m, ag_2:c, ag_3:mc
20	$apply(ag_4, raa, c)$	ag_1:m, ag_2:c, ag_3:mc
21	$apply(ag_5, raa, c)$	ag_1:m, ag_2:c, ag_3:mc
22	$accept\_quote(ag_3, ag_1, book_A, 23)$	ag_1:m, ag_2:c, ag_3:mc
23	$assign(raa, ag_4, c)$	ag_1:m, ag_2:c, ag_3:mc
24	$send\_goods(ag_1, iServer, book_A, `1100101')$	ag_1:m, ag_2:c, ag_3:mc, ag_4:c
25	$withdraw(ag_3, raa, c)$	ag_1:m, ag_2:c, ag_3:mc, ag_4:c
28	$send\_EPO(ag_3, iServer, book_A, 13)$	ag_1:m, ag_2:c, ag_3:mc, ag_4:c
32	$timeout(book_A)$	ag_1:m, ag_2:c, ag_3:mc, ag_4:c
33	$withdraw(ag_1, raa, m)$	ag_1:m, ag_2:c, ag_3:m, ag_4:c
34	$present\_quote(ag_3, ag_2, music_A, 40)$	ag_2:c, ag_3:m, ag_4:c
43	$present\_quote(ag_3, ag_4, music_B, 40)$	ag_2:c, ag_3:m, ag_4:c
44	$withdraw(ag_3, raa, m)$	ag_2:c, ag_3:m, ag_4:c
45	$accept\_quote(ag_4, ag_3, music_B, 40)$	ag_2:c, ag_3:m, ag_4:c
46	$apply(ag_4, raa, m)$	ag_2:c, ag_3:m, ag_4:c
47	$assign(raa, ag_4, m)$	ag_2:c, ag_3:m, ag_4:c
48	$accept\_quote(ag_2, ag_3, music_A, 40)$	ag_2:c, ag_3:m, ag_4:mc
49	$send\_EPO(ag_4, iServer, music_B, 40)$	ag_2:c, ag_3:m, ag_4:mc
50	$request\_quote(ag_2, ag_3, music_A)$	ag_2:c, ag_3:m, ag_4:mc
52	$present\_quote(ag_3, ag_2, music_A, 44)$	ag_2:c, ag_3:m, ag_4:mc
55	$timeout(music_B)$	ag_2:c, ag_3:m, ag_4:mc
		ag_2:c, ag_3:$\mathtt{s}m$, ag_4:mc

discharged — see axiom (18). ag_3 made no further attempts to discharge its obligation by the time the transaction over $book_A$ ended. Consequently, the timeout that took place at time-point 32, signalling the end of the transaction over $book_A$, initiated a sanction for ag_3. More precisely, ag_3 lost the role of consumer (see axiom (25)) because it did not comply with its obligation to send an EPO, *and* it violated earlier the prohibition to accept ag_1's quote — at time-point 22 ag_3 was forbidden to accept the quote, although empowered to accept it, because ag_3 could not afford the associated price (see axiom (16)).

During the negotiation over $book_A$ between ag_1 and ag_3, two agents, ag_4 and ag_5, attempted to enter NetBill by applying for the consumer role — see time-points 20 and 21 respectively. Both applications were valid (see axiom (27)) because ag_4 and ag_5 did not occupy the consumer role, did not have pending applications for that role, satisfied the conditions of that role, expressed in this example by axiom (26), and were not banned from NetBill. Moreover, due to axiom (28), the role assigning authority *raa* became empowered to assign them the consumer role because the constraint that the number of consumers is at most twice the number of merchants would not be violated if ag_4 and ag_5 were assigned the consumer role. At time-point 23 *raa* chose to assign the consumer role to ag_4, although it did not assign that role to ag_5.

After the end of the transaction over $book_A$ ag_1 successfully withdrew from the merchant role and, effectively, withdrew from NetBill, since ag_1 did not occupy any other role. ag_1's withdrawal was successful because ag_1 had no pending quotes or obligations (see axiom (30)).

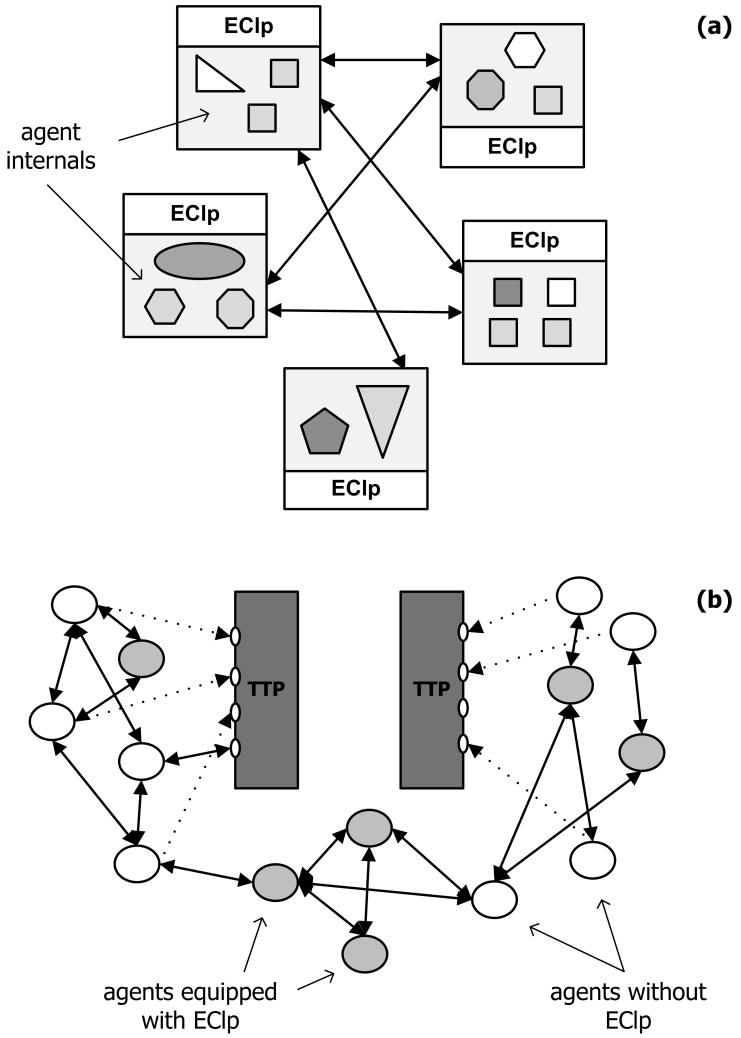

Figure 2. Run-Time Configurations: Solid arrows indicate agent communication and dotted arrows indicate request for information concerning the protocol state.

At time-points 34 and 43 ag_3 initiated respectively two negotiation threads: one concerning $music_A$ and ag_2, and one concerning $music_B$ and ag_4. At time-point 44 ag_3 attempted to withdraw from the merchant role. This attempt was unsuccessful because ag_3 had two pending quotes — the ones mentioned above — at the time. Both quotes were eventually accepted (see time-points 45 and 48); however, only one contract was established, that between ag_3 and ag_4 regarding $music_B$. The quote concerning $music_A$ had expired at time-point 48 and, therefore, the acceptance of ag_2 did not establish a contract. At time-point 49 ag_4 sent an EPO to *iServer* concerning the transaction over $music_B$, thus discharging its obligation to pay (see axiom (18)). ag_3, however, did not discharge its obligation to send the goods. Consequently, the timeout that took place at time-point 55, signalling the end of the transaction over $music_B$, initiated a time period for which ag_3 was suspended from the merchant role (indicated in Table V by $\mathtt{s}m$).

The example narrative displayed in Table V includes a number of events that took place during the negotiation concerning $music_B$: ag_4 applied for, and was eventually assigned, the role of merchant, and a negotiation thread was initiated by ag_2 concerning $music_A$ and ag_3.

6.2. Run-time Configurations

A protocol state, including the powers, permissions, obligations, sanctions and roles that are associated with each agent at each time, computed by the ECIp implementation, may be publicised at run-time to (a subset of) the participants of a protocol, or their designers. Such run-time services may be provided by a central server or in various distributed configurations. Each protocol participant, for example, may have available an ECIp module in order to compute its powers, permissions, obligations, and so on — see Figure 2(a). In another setting, some protocol participants may have direct access to the protocol state by means of a local ECIp module, whereas other agents may rely on trusted third parties (TTP)s for such information — see Figure 2(b). A further discussion of the possible run-time configurations, including the issues that arise in each configuration (such as the policy followed by a TTP for the disclosure of the protocol state), will be presented elsewhere.

7. Discussion

Several approaches have been proposed in the literature for the specification of open MAS — see, for example, [4, 5, 42, 64, 3, 75, 6, 74, 41, 32, 14, 11, 35, 77]. An influential approach has been the work of Moses, Shoham, Tennenholtz and colleagues [49, 50, 67, 68, 73, 25]. These researchers focus on the specification of 'social laws' that govern the behaviour of the members of 'artificial social systems'. In brief, a social law is a set of prohibitions that, if respected, enable agents to co-exist in a shared environment and pursue their goals. Social laws do not express any other normative relations apart from permissions/prohibitions. This is probably due to the fact that the applications that have been studied in the context of the artificial social systems approach (for example, mobile robots moving along a two-dimensional grid [68, Section 2] or a circular automated assembly line [25, Section 3.1]) can be mainly described in terms of physical actions rather than communicative actions, and *brute facts* rather than institutional facts [62, 61]; being in physical possession of an object, for example, is a brute fact (it can be observed), whereas being the owner of that object is an institutional fact. It is assumed, moreover, that agents will be implemented in such a way that all social laws are respected. The question of what happens when a social law is not respected has received little attention.

The artificial social systems approach is not supported by a software implementation for automated reasoning regarding the social laws of a system. In what follows we focus on approaches for the specification of open MAS that have direct routes to implementation and offer computational support.

An approach supported by automated reasoning tools is the well-known work of Minsky and colleagues on 'Law-Governed Interaction' (LGI) [44, 48, 85, 46]. LGI is an abstract regulatory mechanism that satisfies the following principles: statefulness, that is, the regulatory mechanism is sensitive to the history of interaction between the regulated components, decentralisation, for scalability, and generality, that is, LGI is not biased to a particular type of law. LGI is an abstraction of a software mechanism called Moses [45, 47] which can be used to regulate distributed systems. Moses employs regimentation devices — controllers (see Section 4.4) — that monitor the behaviour of agents, block the performance of forbidden actions and enforce compliance with obligations. Laws in Moses are written in pure Prolog or Java.

Esteva and colleagues [22, 23, 21, 20, 57, 24, 30] have devised a specification language to specify open MAS as electronic institutions (e-institutions). The basic components of an e-institution include those of role (standardised pattern of behaviour), dialogic framework (prescribing the agent interactions), scene (expressing sub-groupings created in the context of a wider system),

and normative rule (the 'rules of the game'). Normative rules specify the permissions and obligations of the members of an e-institution.

Software tools for computational support of the e-institution specification language have been developed. Islander [20], for instance, is an integrated development environment for specifying e-institutions. A verification module is implemented which checks for, among other things, 'norm consistency'. A rule-based system [30] has also been developed for executing a set of normative rules with the aim of providing run-time services, such as the computation of the permissions and obligations of the agents current at each state. A precise equivalence, however, between (fragments of) the formalism used for verification of 'norm consistency' with that used for supporting run-time activities, has not been established. Consequently, it is not possible to, say, verify a system specification for 'norm consistency' and then translate that specification to an equivalent one for the provision of run-time services.

Another closely related line of research is the work of Singh and colleagues on commitment protocols [69, 70, 78, 71]. In this context 'commitment' can be seen as an obligation directed from one agent to another. Commitment protocols have been formalised in various ways, giving different types of semantics to the concept of commitment. Yolum and Singh [81,82], for instance, have specified commitment protocols with the use of a subset of Shanahan's 'full version' of Event Calculus (EC) [65] — in this case NetBill was the running example. All 'operations' on commitments — *creating*, *discharging*, *cancelling*, *releasing*, *delegating*, *assigning* commitments — were formalised with the use of EC axioms. Moreover, an abductive EC planner [66] was employed in order to compute planning queries regarding the EC specifications of commitment protocols.

Singh and colleagues have also used the $C+$ language [31], an action formalism with explicit transition systems semantics, to express commitment protocols — see [8, 16, 17] for a few recent papers. (As mentioned earlier in the paper, in [2] we presented a comparison of $C+$ and EC with respect to open MAS specification.) Like EC, $C+$ was employed to express all operations on commitments as well as the effects of the agents' actions. The Causal Calculator [31], a software implementation using satisfiability (SAT) solvers to compute planning and narrative assimilation queries regarding $C+$ formalisations, was employed to execute commitment protocol specifications.

It is difficult to see how an interaction protocol for open MAS (such as a protocol for e-commerce, negotiation, dispute resolution or voting) can be specified simply in terms of commitments. The concept of institutional power is at least as important when specifying a protocol and the meaning of protocol actions. In NetBill, for example, it is necessary to identify the circumstances in which a consumer has the 'legal capacity' or institutional power to accept a quote and, therefore, establish a contract. Similarly, it is important to specify the conditions in which an agent is empowered to initiate proceedings against the other contracting party, the circumstances in which an agent is empowered to apply for participation in NetBill, and so on. It is not difficult to see that there are other protocols in which a specification simply in terms of commitments is inappropriate.

Commitment protocols have also been studied by Colombetti and colleagues — see [79, 27, 28, 26] for a few recent publications. In this line of work, unlike the approaches reviewed so far, the concept of institutional power is represented. The formalisation of this concept, however, is unclear. Although being empowered is a necessary condition for performing operations on commitments in some cases, there are examples in which an agent may successfully modify a set of commitments — create a commitment in which the agent is the creditor, cancel a commitment, etc — without having the institutional power to modify these commitments. Moreover, exercising the institutional power to perform an action is not always effective in creating or modifying a

set of institutional facts (even if the action is not blocked, say by a regimentation device). In the formalisation presented in [26], for example, an agent Ag is empowered to assign any role to some other agent. Exercising the power to assign role R, however, has no effects if R is not an existing role of the protocol in question.

In the commitment protocol specifications there are examples in which an agent is empowered to decrease the 'trust' in an agent [26]. Unless 'trust' is somehow seen as an institutional fact, this is not a meaningful instance of institutional power.

Colombetti and colleagues have very recently [26] employed a dialect of EC to express commitment protocols and a SAT-based implementation [51] to execute these protocols. However, the computation of the supported tasks — narrative assimilation and planning — is inefficient because the complete protocol history is recorded in the protocol states. Another implementation route has been the use of a model checker for proving properties of a commitment protocol specification [79]. No comparison between the two implementation routes, in terms of representation language expressiveness or computational efficiency, has been presented.

Related work from the field of computational organisation theory includes the work of Fox and colleagues on enterprise modelling [29, 33]. In this approach a multi-agent organisation is viewed as a set of agents playing roles in which they are acting to achieve specific goals according to various constraints defining the 'rules of the game'. The rules are formalised with the use of a dialect of the Situation Calculus [52], an implementation of which supports the tasks of planning and narrative assimilation. (A comparison of the Situation Calculus and the action formalisms $C+$ and EC, used for commitment protocol specification, may be found in [51], for example.) Fox and colleagues identify and represent several normative relations of the members of an organisation, such as obligation, authority and empowerment. The representation of 'empowerment', however, is not clear. In some cases 'empowerment' seems to coincide with the concept of institutional power presented in our paper, in the sense that an empowered agent may create a set of institutional facts, such as the establishment of a contract with a third party, while in other cases 'empowerment' seems to coincide with permission, in the sense that performing an action will not be penalised.

The main contribution of this paper then, and a key difference between our work and related research, is that we provide an explicit, unambiguous specification of institutional powers, and distinguish between institutional power, permission/obligation and physical capability. To illustrate this distinction we presented in this paper various example formalisations of physical capability, institutional power, permission and obligation. Moreover, we presented example formalisations of sanctions that deal with the performance of forbidden actions and violation of obligations. Consequently we do not have to rely exclusively on regimentation mechanisms.

Another difference of our work from related MAS research is that we distinguish between an open MAS and the corresponding role-management process — applying for membership, withdrawing from a MAS, and so on. To illustrate this distinction, we presented an example formalisation of a role-management protocol for NetBill.

Our protocol specifications are executable. In Section 6 we presented an execution of NetBill and the associated role-management protocol. A protocol specification may be executed at run-time, for example, in order to compute the agents' powers, permissions, obligations, and so on, with the aim of informing the agents' decision-making. The software implementations discussed in this section — with the exception of Moses — operate under the assumption that the domain of each variable is finite and known from the outset. For example, we would have to know from the outset all possible participants of NetBill, all possible items for which a merchant may present quotes, and so on. A run-time modification of the domain of a variable requires a re-compilation of the specification. The procedure of re-compiling a specification can be very

time-consuming, making the associated software implementation unsuitable for the provision of run-time services. EClp does not require that the domain of a variable is finite or known from the outset. Consequently it is not required to re-compile a specification at run-time.

Our EC implementation has proven to be faster than other action language implementations we have tried (see [2] for comparisons), and can be further optimised by incorporating various techniques proposed in the literature for narrative assimilation in EC ([7] for example). Our EC implementation, however, does not offer facilities for proving properties of a specification or planning. A direction for further work is to employ a single formalism for narrative assimilation and proving properties. Some first steps are reported in [12] which investigates methods for efficient EC-like query evaluation for (a subset of) the $C+$ and $nC+$ languages, and for integrating action descriptions in these languages with standard model checking systems (specifically NuSMV [9]). Normative system properties expressed in temporal logics such as Computation Tree Logic (CTL) can then be verified by means of standard model checking techniques on a transition system defined using the $nC+$ language.

Our executable specifications may be classified as 'static', in the sense that there is no support for their run-time modification. In some open systems, however, environmental, social or other conditions may favour, or even require, specifications modifiable during the system execution. Consider, for instance, the case in which the participants of NetBill require to change the conditions of the consumer role, or the allowed ratio between consumers and merchants. We are currently working towards 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. According to this infrastructure, at any time during the protocol execution an agent may attempt to modify the protocol specification by initiating a 'meta-protocol' for deciding about — arguing, negotiating, voting for/against — a proposed modification.

References

1. A. Artikis, M. Sergot, and J. Pitt. An executable specification of a formal argumentation protocol. *Artificial Intelligence*, 171(10–15):776–804, 2007.
2. A. Artikis, M. Sergot, and J. Pitt. Specifying norm-governed computational societies. *ACM Transactions on Computational Logic*, 10(1), 2009. Retrieved July 6, 2008, from http://www.acm.org/pubs/tocl/accepted/304artikis.pdf.
3. A. Bandara. *A Formal Approach to Analysis and Refinement of Policies*. PhD thesis, Imperial College London, 2005.
4. G. Boella and L. van der Torre. Security policies for sharing knowledge in virtual communities. *IEEE Transactions on Systems, Man and Cybernetics*, 36(3):439–450, 2006.
5. G. Boella and L. W. N. van der Torre. The ontological properties of social roles in multi-agent systems: Definitional dependence, powers and roles playing roles. *Artificial Intelligence and Law*, 15(3):201–221, 2007.
6. J. Bradshaw, A. Uszok, R. Jeffers, N. Suri, P. Hayes, M. Burstein, A. Acquisti, B. Benyo, M. Breedy, M. Carvalho, D. Diller, M. Johnson, S. Kulkarni, J. Lott, M. Sierhuis, and R. Van Hoof. Representation and reasoning about DAML-based policy and domain services in KAoS. In J. Rosenschein, T. Sandholm, M. Wooldridge, and M. Yokoo, editors, *Proceedings of Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 835–842. ACM Press, 2003.
7. L. Chittaro and A. Montanari. Efficient temporal reasoning in the cached event calculus. *Computational Intelligence*, 12(3):359–382, 1996.
8. 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.
9. A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *Proceedings of International Conference on Computer-Aided Verification (CAV 2002), Copenhagen, July 2002*, LNCS 2404. Springer, 2002. See http://nusmv.irst.itc.it.

10. K. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 293–322. Plenum Press, 1978.
11. O. Cliffe, M. De Vos, and J. Padget. Answer set programming for representing and reasoning about virtual institutions. In *Proceedings of Workshop on Computational Logic in Multi-Agent Systems (CLIMA)*, LNCS 4371. Springer, 2007.
12. R. Craven. *Execution Mechanisms for the Action Language $\mathcal{C}$+*. PhD thesis, University of London, September 2006.
13. R. Craven and M. Sergot. Agent strands in the action language nC+. *Journal of Applied Logic*, 6(2):172–191, June 2008.
14. M. Dastani, V. Dignum, and F. Dignum. Role-assignment in open agent societies. In *Proceedings of Conference on Autonomous Agents and Multi-Agent Systems*, pages 489–496. ACM, 2003.
15. C. Dellarocas. Reputation mechanisms. In T. Hendershott, editor, *Handbook on Economics and Information Systems*. Elsevier Publishing, 2006.
16. N. Desai and M. Singh. A modular action description language for protocol composition. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 2007.
17. N. Desai and M. Singh. Checking correctness of business contracts via commitments. In *Proccedings of International Conference on Autonomous Agents and Multi-Agent Systems*, 2008.
18. 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.
19. H.-P. Dommel and J. J. Garcia-Luna-Aceves. Floor control for multimedia conferencing and collaboration. *Multimedia Systems*, 5(1):23–38, 1997.
20. M. Esteva, D. de la Cruz, and C. Sierra. ISLANDER: an electronic institutions editor. In C. Castelfranchi and L. Johnson, editors, *Proceedings of Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1045–1052. ACM Press, 2002.
21. M. Esteva, J. Padget, and C. Sierra. Formalizing a language for institutions and norms. In J.-J. Meyer and M. Tambe, editors, *Intelligent Agents VIII: Agent Theories, Architectures, and Languages*, LNAI 2333, pages 348–366. Springer, 2002.
22. M. Esteva, J. Rodriguez-Aguilar, J. Arcos, C. Sierra, and P. Garcia. Institutionalising open multi-agent systems. In E. Durfee, editor, *Proceedings of the International Conference on Multi-agent Systems (ICMAS)*, pages 381–382. IEEE Press, 2000.
23. 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.
24. M. Esteva, J. Rodríguez-Aguilar, C. Sierra, and W. Vasconcelos. Verifying norm consistency in electronic institutions. In *Proceedings of the AAAI-04 Workshop on Agent Organizations: Theory and Practice*, pages 8–14, 2004.
25. D. Fitoussi and M. Tennenholtz. Choosing social laws for multi-agent systems: minimality and simplicity. *Artificial Intelligence*, 119(1-2):61–101, 2000.
26. N. Fornara and M. Colombetti. *Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, chapter Formal specification of artificial institutions using the event calculus. IGI Global, to appear in 2009.
27. N. Fornara, F. Viganò, and M. Colombetti. Agent communication and artificial institutions. *Autonomous Agents and Multi-Agent Systems*, 14:121–142, 2007.
28. N. Fornara, F. Viganò, M. Verdicchio, and M. Colombetti. Artificial institutions: A model of institutional reality for open multiagent systems. *Artificial Intelligence and Law*, 16:89–105, 2008.
29. M. Fox, M. Barbuceanu, M. Grüninger, and J. Lin. An organizational ontology for enterprise modeling. In M. Prietula, K. Carley, and L. Gasser, editors, *Simulating Organizations: Computational Models for Institutions and Groups*, pages 131–152. AAAI Press/The MIT Press, 1998.
30. A. García-Camino, P. Noriega, and J. Rodríguez-Aguilar. Implementing norms in electronic institutions. In *Proceedings of the Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 667–673. ACM Press, 2005.
31. E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1–2):49–104, 2004.
32. D. Grossi, F. Dignum, and J.-J. Ch. Meyer. A formal road from institutional norms to organizational structures. In E. Durfee, M. Yokoo, M. Huhns, and O. Shehory, editors, *Proceedings of Conference on Autonomous Agents and Multi-Agent Systems*, pages 616–623, 2007.

33. M. Grüninger and M. Fox. The role of competency questions in enterprise engineering. In *Proceedings of the IFIP WG5.7 Workshop on Benchmarking-Theory and Practice*, 1994.
34. C. Hewitt. Open information systems semantics for distributed artificial intelligence. *Artificial Intelligence*, 47:79–106, 1991.
35. J. Hubner, J. Sichman, and O. Boissier. Developing organised multiagent systems using the MOISE$^+$ model: programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering*, 1(3–4):370–395, 2007.
36. A. Jones. Roles. In A. Jones and C. Krogh, editors, *Deliverable D2 of ALFEBIITE EU-Project (IST-1999-10298)*, pages 52–55, 2001.
37. 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.
38. A. Jones and M. Sergot. A formal characterisation of institutionalised power. *Journal of the IGPL*, 4(3):429–445, 1996.
39. 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 Multi-Agent Systems*, 7(1–2):179–189, 2003.
40. R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–96, 1986.
41. L.Kagal and T. Finin. Modeling communicative behavior using permissions and obligations. *Journal of Autonomous Agents and Multi-Agent Systems*, 14(2):187–206, 2006.
42. A. Lomuscio and M. Sergot. Deontic interpreted systems. *Studia Logica*, 75(1):63–92, 2003.
43. D. Makinson. On the formal representation of rights relations. *Journal of Philosophical Logic*, 15:403–425, 1986.
44. N. Minsky. The imposition of protocols over open distributed systems. *IEEE Transactions on Software Engineering*, 17(2):183–195, 1991.
45. N. Minsky. *Law-Governed Interaction (LGI): A Distributed Coordination and Control Mechanism (An Introduction and a Reference Manual)*, 2005. Retrieved October 24, 2008, from http://www.moses.rutgers.edu/documentation/manual.pdf.
46. N. Minsky. Decentralised regulation of distributed systems: Beyond access control. Submitted for publication. Retrieved October 24, 2008, from http://www.cs.rutgers.edu/~minsky/papers/IC.pdf, 2008.
47. N. Minsky and T. Murata. On manageability and robustness of open multi-agent systems. In *Software Engineering for Multi-Agent Systems II, Research Issues and Practical Applications*, LNCS 2940, pages 189–206. Springer, 2004.
48. 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.
49. Y. Moses and M. Tennenholtz. On computational aspects of artificial social systems. In *Proceedings of Workshop on Distributed Artificial Intelligence (DAI)*, pages 267–284, 1992.
50. Y. Moses and M. Tennenholtz. Artificial social systems. *Computers and Artificial Intelligence*, 14(6):533–562, 1995.
51. E. Mueller. *Commonsense Reasoning*. Morgan Kaufmann, 2006.
52. J. Pinto and R. Reiter. Temporal reasoning in logic programming: a case for the situation calculus. In D. Warren, editor, *Proceedings of Conference on Logic Programming*, pages 203–221. MIT Press, 1993.
53. J. Pitt, L. Kamara, M. Sergot, and A. Artikis. Voting in multi-agent systems. *Computer Journal*, 49(2):156–170, 2006.
54. J. Pitt, A. Mamdani, and P. Charlton. The open agent society and its enemies: a position statement and research programme. *Telematics and Informatics*, 18(1):67–87, 2001.
55. H. Prakken. Formalising Robert's rules of order. Technical Report 12, GMD – German National Research Center for Information Technology, 1998.
56. 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.
57. J. Rodriguez-Aguilar and C. Sierra. Enabling open agent institutions. In K. Dautenhahn, A. Bond, L. Canamero, and B. Edmonds, editors, *Socially Intelligent Agents: Creating relationships with computers and robots*, pages 259–266. Kluwer Academic Publishers, 2002.
58. J. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. The MIT Press, 1994.

59. J. Rosenschein and G. Zlotkin. Designing conventions for automated negotiation. In N. Huhns and M. Singh, editors, *Readings in Agents*, pages 353–370. Morgan Kaufmann, 1998.
60. J. Sabater and C. Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24(1):33–60, 2005.
61. J. Searle. *Speech Acts*. Cambridge University Press, 1969.
62. J. Searle. What is a speech act? In A. Martinich, editor, *Philosophy of Language*, pages 130–140. Oxford University Press, third edition, 1996.
63. M. Sergot. Modelling unreliable and untrustworthy agent behaviour. In B. Dunin-Keplicz, A. Jankowski, A. Skowron, and M. Szczuka, editors, *Proceedings of Workshop on Monitoring, Security, and Rescue Techniques in Multiagent Systems (MSRAS)*, Advances in Soft Computing, pages 161–178. Springer-Verlag, 2004.
64. M. Sergot and R. Craven. The deontic component of action language $nC+$. In L. Goble and J-J. Ch. Meyer, editors, *Deontic Logic in Computer Science (DEON)*, LNAI 4048, pages 222–237. Springer, 2006.
65. M. Shanahan. The event calculus explained. In M. Wooldridge and M. Veloso, editors, *Artificial Intelligence Today*, LNAI 1600, pages 409–430. Springer, 1999.
66. M. Shanahan. An abductive event calculus planner. *Journal of Logic Programming*, 44:207–239, 2000.
67. Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In W. Swartout, editor, *Proceedings of Conference on Artificial Intelligence (AAAI)*, pages 276–281. The AAAI Press/ The MIT Press, 1992.
68. Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, 73(1-2):231–252, 1995.
69. M. Singh. Agent communication languages: rethinking the principles. *IEEE Computer*, 31(12):40–47, 1998.
70. M. Singh. An ontology for commitments in multiagent systems: towards a unification of normative concepts. *Artificial Intelligence and Law*, 7(1):97–113, 1999.
71. 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.
72. M. Sirbu. Credits and debits on the Internet. *IEEE Spectrum*, 34(2):23–29, 1997.
73. M. Tennenholtz. On computational social laws for dynamic non-homogeneous social structures. *Journal of Experimental and Theoretical Artificial Intelligence*, 7:379–390, 1995.
74. A. Uszok, J. Bradshaw, R. Jeffers, M. Johnson, A. Tate, J. Dalton, and S. Aitken. Policy and contract management for semantic web services. In *Proceedings of the AAAI Spring Symposium on Semantic Web Services*, 2004.
75. A. Uszok, J. Bradshaw, J. Lott, M. Breedy, L. Bunch, P. Feltovich, M. Johnson, and H. Jung. New developments in ontology-based policy management: Increasing the practicality and comprehensiveness of KAoS. In *Proceedings of Workshop on Policies for Distributed Systems and Networks*, pages 145–152. IEEE Computer Society, 2008.
76. R. van Eijk, F. de Boer, W. van der Hoek, and J.-J. Meyer. Open multi-agent systems: agent communication and integration. In N. Jennings and Y. Lesperance, editors, *Intelligent Agents VI: Agent Theories, Architectures, and Languages*, LNCS 1757, pages 218–232. Springer, 2000.
77. J. Vazquez-Salceda, V. Dignum, and F. Dignum. Organizing multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 11(3):307–360, 2005.
78. M. Venkatraman and M. Singh. Verifying compliance with commitment protocols. *Journal of Autonomous Agents and Multi-Agent Systems*, 2(3):217–236, 1999.
79. F. Viganò and M. Colombetti. Symbolic model checking of institutions. In *Proceedings of Conference on Electronic commerce*, pages 35–44. ACM, 2007.
80. E. Werner. The design of multi-agent systems. In E. Werner and Y. Demazeau, editors, *Decentralized A.I.*, volume 3, pages 3–30. Elsevier Science Publishers, 1992.
81. P. Yolum and M. Singh. Flexible protocol specification and execution: applying event calculus planning using commitments. In C. Castelfranchi and L. Johnson, editors, *Proceedings of Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 527–535. ACM Press, 2002.
82. P. Yolum and M. Singh. Reasoning about commitments in the event calculus: An approach for specifying and executing protocols. *Annals of Mathematics and Artificial Intelligence*, 42(1–3):227–253, 2004.
83. B. Yu. Distributed reputation management for electronic commerce. *Computational Intelligence*, 18(4):535–549, 2002.
84. F. Zambonelli, N. Jennings, and M. Wooldridge. Organisational rules as an abstraction for the analysis and design of multi-agent systems. *International Journal of Software Engineering and Knowledge Engineering*, 11(3):303–328, 2001.

85. W. Zhang, C. Serban, and N. Minsky. Establishing global properties of multi-agent systems via local laws. In D. Weyns, editor, *Environments for Multiagent Systems III*, volume LNAI 4389. Springer, 2007.