Specifying Norm-Governed Computational Societies

Alexander Artikis, Marek Sergot and Jeremy Pitt
Imperial College London, SW7 2BZ, UK
{a.artikis, m.sergot, j.pitt}@imperial.ac.uk

Abstract

Electronic markets, dispute resolution and negotiation protocols are three types of application domains that can be viewed as open agent societies. Key characteristics of such societies are agent heterogeneity, conflicting individual goals and unpredictable behaviour. Members of such societies may fail to, or even choose not to, conform to the norms governing their interactions. It has been argued that systems of this type should have a formal, declarative, verifiable, and meaningful semantics. We present a theoretical and computational framework being developed for the executable specification of open agent societies. We adopt an external perspective and view societies as instances of normative systems. In this paper we demonstrate how the framework can be applied to specifying and executing a contract-net protocol. The specification is formalised in two action languages, the $\mathcal{C}+$ language and the Event Calculus, and executed using respective software implementations, the Causal Calculator and the Society Visualiser. We evaluate our executable specification in the light of the presented case study, discussing the strengths and weaknesses of the employed action languages for the specification of open agent societies.

1. INTRODUCTION

A particular kind of Multi-Agent System (MAS) is one where the members are developed by different parties and have conflicting goals. A key characteristic of this kind of MAS, due to the globally inconsistent goals of its members, is the high probability of non-conformance to the specifications that govern the members' interactions. A few examples of this type of MAS are negotiation protocols [Smith and Davis 1978; Rosenschein and Zlotkin 1994], dispute resolution protocols [Brewka 2001; Prakken 2005], rules of procedure [Prakken and Gordon 1999], electronic marketplaces [Sirbu 1997], Virtual Enterprises [Hardwick and Bolton 1997; Cevenini 2003], Virtual Organisations [Foster et al. 2001], and digital media rights management [Bing 1998]. Multi-agent systems of this type are often classified as 'open'.

It has been argued that many practical applications in the future will be realised in terms of 'open agent systems' of this sort. Not surprisingly, there is growing interest in the MAS community in such systems. Several researchers, for example, view open agent systems as computational organisations and use organisational abstractions to specify them. Esteva et al. [2000; 2001; 2002; 2002], Rodriguez-Aguilar and Sierra [2002] have devised a specification language to specify open agent systems as electronic institutions (or 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 'obligations' and 'commitments' of the members of an e-institution.

Zambonelli et al. [2001a; 2001; 2001b; 2003] state that approaches like the AALAADIN meta-model of multi-agent organisations [Ferber and Gutknecht 1998; 2000] and the Gaia methodology for agent-oriented analysis and design [Wooldridge et al. 1999; 2000] view computational organisations simply as collections of roles and do not incorporate the necessary notion of 'organisational rule'. In their extension of Gaia, Zambonelli et al. [2001a; 2003] express organisational rules, that is, global constraints prescribing the behaviour of the members of an organisation, with the use of two alternative formalisms: (i) a subset of a first-order temporal logic [Manna and Pnueli 1992; 1995], and (ii) regular expressions, a notation based on the FUSION notation for operation schemata [Coleman et al. 1994]. Fox et al. [1998] express organisational rules with the use of a dialect [Pinto and Reiter 1993] of the Situation Calculus, placing emphasis on the concepts of 'permission', 'right', 'authority', and so on.

Closely related work in multi-agent systems includes the work of Moses and Tennenholtz [1992; 1995], Shoham and Tennenholtz [1992; 1995], Tennenholtz [1995], Fitoussi and Tennenholtz [2000], who 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. Minsky and Ungureanu [2000] propose a mechanism for coordination in open agent systems, called 'law-governed interaction', which is based on the following principles: (i) coordination policies need to be enforced, (ii) the enforcement needs to be decentralised (in order to avoid a single point of failure), (iii) coordination policies need to be explicitly formulated rather than being implicitly described in the agents' internals, and (iv) a policy should be deployed without exacting a cost from the agents not subject to it. Coordination policies specify, amongst other things, the 'enforced obligations' of the agents, that is, obligations that are either discharged by the agents or enforced by a dedicated regimentation device.

In another line of research on open agent systems, Singh [1998; 2000] argues that the semantics for Agent Communication Languages (ACLs) in such systems must be *formal*, *declarative* (the semantics should describe *what* rather than *how*), *verifiable* (it should be possible to determine whether an agent is acting according to the system specification) and *meaningful* (the semantics should be based on some intuitive appreciation of the system under consideration). Moreover, he and Colombetti [2000] argue that the notion of 'social commitment' is a sound basis for the ACL semantics in open agent systems, as opposed to mentalistic notions such as belief, desire and intention.

Within this general context, this paper is concerned with the presentation of *executable specifications* of open agent systems. We here consider a multi-agent system as open if it exhibits the following characteristics:

(1) The internal architectures of the members are not publicly known.

(2) Members do not necessarily share a notion of global utility [Rosenschein and

Zlotkin 1994].

(3) The behaviour and the interactions of the members cannot be predicted in advance [Hewitt 1991].

The first of these characteristics implies that an open agent system may be composed of agents with different internal architectures. Therefore, we will treat open agent systems 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 agent system may fail to, or even choose not to, conform to the specifications (of that system) in order to achieve their individual goals (this is what Minsky and Ungureanu [2000] referred to as *inadvertent* and *malicious* violations respectively). And further, open agent systems are always subject to unanticipated outcomes in their interactions [Hewitt 1991].

We restrict attention to open agent systems in which the behaviour of the members is governed by a set of *social laws*. Moreover, each member in these systems occupies at least one *social role*. Drawing an analogy from human societies, we call the multi-agent systems of this type *open agent societies*, and use the terms 'open agent society' and 'open, norm-governed computational society' interchangeably.

In constructing executable specifications of open agent societies [Artikis et al. 2002; Artikis et al. 2003b; 2003a; Artikis 2003], we adopt a bird's eye view of these systems, as opposed to an agent's own perspective whereby it reasons about how it should act. Furthermore, we view agent systems as instances of *normative systems* [Jones and Sergot 1993]. A feature of this type of system is that actuality, what is the case, and ideality, what ought to be the case, do not necessarily coincide. Therefore, it is essential to specify what is permitted, prohibited, and obligatory, and perhaps other more complex normative relations (such as duty, right, privilege, authority, ...) that may exist between the agents. Amongst these relations, we place considerable emphasis on the representation of *institutionalised power* [Searle 1969; Jones and Sergot 1996] — a standard feature of any norm-governed system whereby designated agents, when acting in specified roles, are empowered by an institution to create specific relations or states of affairs (such as when an agent is empowered by an institution to award a contract and thereby create a bundle of normative relations between the contracting parties).

We encode specifications of open agent societies in executable action languages. In this paper we show how two such languages from the field of Artificial Intelligence (AI) may be used to express these specifications: the $\mathcal{C}+$ language [Giunchiglia et al. 2001; Giunchiglia et al. 2004] and the Event Calculus (EC) [Kowalski and Sergot 1986]. The $\mathcal{C}+$ language, notably when used with its associated software implementation, the Causal Calculator (CCALC), already supports a wide range of computational tasks of the kind that we wish to perform on society specifications. A major attraction of $\mathcal{C}+$ compared with other action languages in AI is its explicit semantics in terms of labelled transition systems, a familiar structure widely used in logic and computer science. EC, on the other hand, does not have an *explicit* transition system semantics, but has the merits of being simple, flexible, and very easily and efficiently implemented for an important class of computational tasks. It thus provides a practical means of implementing an executable society specifi-

cation. The Society Visualiser (SV), described later in the paper, is a software implementation that supports computational tasks on society specifications formulated in EC. The relative advantages and disadvantages of these action formalisms are discussed in the concluding sections of the paper. A third candidate formalism is the language $(\mathcal{C}+)^{++}$ presented in [Sergot 2004a]. $(\mathcal{C}+)^{++}$ is an extended form of $\mathcal{C}+$ specifically designed for modelling the normative and institutional aspects of MAS. However, since the development of $(\mathcal{C}+)^{++}$ took place in parallel with the methods presented in this paper, we leave its presentation to a separate paper.

This paper is structured as follows. First, we present a theoretical framework for specifying open agent societies. More precisely, we present a specification of the social laws (or *social constraints*) and social roles of an open agent society. Second, we review the $\mathcal{C}+$ language and the associated implementation, CCALC. We illustrate the use of $\mathcal{C}+$ and the theoretical framework by specifying a variation of the Contract-Net Protocol (CNP) [Smith and Davis 1978; Smith 1980; Davis and Smith 1983] and executing this specification with CCALC. Third, we present the dialect of EC that we use and its implementation in SV, and then an EC specification of the CNP, and its execution using SV. In [Artikis et al. 2002] we specified and executed the CNP with the use of EC and SV whereas in [Artikis et al. 2003b] we specified and executed the CNP with the use of the $\mathcal{C}+$ language and CCALC. Here, we present an updated and more detailed account of both specifications and the respective executions. Finally, we evaluate the executable specification in the light of the presented case study, identifying strengths and weaknesses of this type of specification, and discuss ways to overcome some of the reported weaknesses.

2. THEORETICAL FRAMEWORK

An open agent society can be expressed in terms of a set of *agents* (the members of a society), a set of *constraints* on a society (norms, and other constraints, such as physical and logical constraints), a set of *roles* that members can play, the *state* of the members and the environment in which they act, a *communication language*, relationships between the members, including *ownership* and *representation* relations, and the *structure* of an open agent society.

There are several approaches in the literature that study these concepts. For example, studies of social structure of MAS may be found in [Werner 1989; Tennenholtz 1995; Esteva et al. 2001]. Some first attempts at formalising representation relations may be found in [Gelati et al. 2002; Gelati et al. 2002; Governatori et al. 2002]. We focus on the specification of social constraints, social roles and social states. These are specified at the design stage of an open agent society. Furthermore, we assume that the specification of social constraints does not change at run-time, that is, during the execution of a society.

We view open agent societies from an external perspective (also referred to as *meta-perspective* by Werner [1992, p.7]). Our specification is based only on externally observable states of affairs and not on the internals of the members. 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.

Two additional assumptions are made:

—We anticipate applications in which agents are members of different societies. However, the analysis that follows does not deal with the issue of multiple societies. We assume that only a single society exists.

—In our view, apart from its members, an open agent society may include other groupings, which we call, following standard usage, *institutions* [Searle 1969; Jones and Sergot 1996; Carley and Gasser 1999; Santos et al. 1997]. Such institutions have their own constraints, roles, communication language, and so on. For simplicity, in this paper we assume that each open agent society includes members of a single institution.

Due to the last two assumptions, in the following sections we do not relativise the specification of social constraints to a particular society or institution within a society. The relaxation of the single-institution and single-society assumptions, as well as the assumption regarding the design-time specification of the social constraints, raise a number of further complications and will be presented in a separate paper.

The following sections describe the specification of the social constraints and roles of an open agent society. Our treatment of social states is discussed in Sections 4 and 8, including the way we execute the specification of an open agent society.

2.1 Social Constraints

We maintain the standard and long established distinction between physical capability, institutionalised power and permission (see, for instance, [Makinson 1986; Jones and Sergot 1996] for illustrations of this distinction). Accordingly, we present a four-level specification of the social constraints of an open agent society, that expresses:

—the physical capabilities,

—institutionalised powers,

—permissions, prohibitions and obligations of the agents;

—the *sanctions* and *enforcement policies* that deal with the performance of forbidden actions and non-compliance with obligations.

The first level of specification concerns the externally observable physical capabilities of a society's members (in a virtual soccer field [Noda et al. 1998], for instance, we may want to express the conditions in which an agent is capable of 'kicking' the ball). The remaining three levels of specification are described next.

2.1.1 *Institutionalised 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* [Searle 1969] — 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). This concept has received considerable attention within the jurisprudential literature, usually under the headings of 'legal power', 'legal capacity' or 'norms of competence', but it is clear that it is not an exclusively *legal* phenomenon — it is a standard feature of all organised interaction.

According to the account given by Jones and Sergot [1996], 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 (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, or banging the table with a wooden mallet counts as declaring a meeting closed).

For the specification of the effects of actions within institutions, it is important — essential — to distinguish between, for example, the act of making an offer and the act by means of which that offer is made (such as sending a letter). Banging the table with a wooden mallet is not the same act as closing a meeting. Indeed, it is only if the table is banged by a person with the institutional power to close the meeting that the meeting is thereby declared closed; the same act performed by an agent without this power has no effect on the status of the meeting (though it may have other effects). In such examples we say that an agent 'has the institutional power' (or just 'power'), or 'is empowered', to close the meeting by means of banging the table with a wooden mallet.

In some circumstances it is awkward or unnecessary to isolate and name all instances of the acts by means of which agents exercise their institutional powers. When describing an auction, for example, it is convenient to say 'agent x made a bid' and let context disambiguate whether we mean by this that the agent performed an action, such as raising its hand, by means of which the making of a bid is signalled, or whether agent x actually made a bid, in the sense that the current bidding price of the item under auction was changed. We find it convenient to 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. So, when we say that 'agent x made a bid y' we mean, by convention, merely that agent x signalled its intention to make a bid y; this act was not necessarily effective in changing the current bidding price. In order to say that the bidding price was actually changed, we say that the action 'agent x made a bid y' was valid: not only did x signal its intention to make bid y, but also x was empowered to make the bid y at that time. Similarly, 'invalid' is used to indicate lack of institutional power: when we say 'agent x made a bid y but that was invalid' we mean that x signalled its intention to make bid y but did not have the institutional power to make that bid at that time (and so the attempt to change the current bidding price was not successful). Differentiating between valid ('meaningful') and invalid ('meaningless') actions is of great importance in the analysis of MAS. In an auction, for example, the auctioneer has to determine which bids are *valid*, and therefore which bids are *eligible* for winning the auction.

We are conscious that this use of the term 'valid' is not ideal and indeed may be inappropriate in some contexts. Terms such as 'valid', 'in order', 'proper' (and 'invalid', 'out of order', 'improper', 'void') have specific meanings in certain 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 — we will stick to the term 'valid' in this paper.

2.1.2 *Permission.* This level of specification of social constraints provides the definitions of permitted, prohibited and obligatory actions. These definitions are application-specific. In some cases, we might want to associate institutional powers with permissions. In some societies, for example, an agent is permitted to perform an action if that agent is empowered to perform that action. According to this definition, an agent is always permitted to exercise its institutional powers. In other societies the relationship is stronger: an agent is permitted to perform an action if *and only if* it is empowered to perform that action. In general, however, there is no standard, fixed relationship between powers and permissions. For example, it is sometimes valuable to *forbid* an agent to perform an action even if it is empowered to perform that action. Similarly, the specification of obligations is application-specific. It is important, however, to maintain consistency of the specification of permissions and obligations on the same society: an agent should not be forbidden and obliged to perform the same action at the same time.

Determining what actions are permitted, prohibited or obligatory enables the classification of the behaviour of individual agents and the society as a whole into categories such as 'social' or 'anti-social', 'acceptable' or 'unacceptable', and so on. For example, the behaviour of an agent might be considered 'anti-social' or 'unacceptable' if that agent performs certain forbidden actions or does not comply with its obligations. Based on the behaviour of the individual agents, it is possible to classify the behaviour of the society as a whole. For example, the state of a society may be considered 'unacceptable' if the majority of its members have not complied with their obligations.

2.1.3 *Enforcement Policies.* This level of specification of social constraints expresses the sanctions and enforcement policies that deal with 'anti-social' or 'unacceptable' behaviour. We are concerned with the following issues: (i) when is an agent sanctioned, and (ii) what is the penalty that the agent has to face (in the case that it does get sanctioned). The specification of both of these issues is also application-specific. As far as the first is concerned, agents may be sanctioned for not complying with their obligations, or they may be sanctioned if they perform forbidden actions. As regards the second issue, penalties can come in many different forms. The house rules of an auction house, for example, may stipulate that bidders who bid out of turn (and are therefore considered 'sanctioned') are no longer empowered to enter other auctions. In different settings, the same type of misbehaviour might create different sanctions. One common type of sanction may be expressed in terms of a social concept such as *bad reputation*.

Sanctions are one means by which an open agent society may discourage 'unacceptable' or 'anti-social' behaviour. Another mechanism is to try to devise additional controls (physical or institutional) that will force agents to comply with their obligations or prevent them from performing forbidden actions. In an automated auction, for example, forbidden (non-permitted) bids may be physically blocked, in the sense that their transmission is disabled, or the specification of a valid bid may be changed to render them ineffective. The general strategy of designing mechanisms to force compliance and eliminate non-permitted behaviour is what Jones and Sergot [1993] termed *regimentation*. Sanctioning mechanisms are required because the opportunities for effective regimentation are usually very limited.

2.2 Social Roles

Being motivated by Jones [2001], Pörn [1977], we associate a social role with a set of *preconditions* that agents must satisfy in order to be eligible to occupy that role, and a set of *constraints* that govern the behaviour of the agents once they occupy that role.

Agents usually participate in a *role-assignment protocol* before entering an open agent society in order to acquire a set of roles that they will occupy while being part of that society. In general, an agent may be assigned a role if the following criteria are met:

—The agent satisfies the role preconditions. It should be possible to determine whether or not an agent satisfies the preconditions of a role without having to access its internals. Agents may acquire certificates, for example, that prove that they satisfy the preconditions of a role.

—The assignment of the role to the agent does not violate the *role-assignment constraints*. These constraints are defined in an application-specific manner — the role-assignment constraints of an auction house, for instance, may require that at most one agent may occupy the role of the auctioneer.

The role constraints, that is, the constraints prescribing the behaviour of an agent occupying a role R, specify the powers, permissions, obligations and sanctions associated with R. The set of role constraints of a role R is a subset of the set of social constraints.

We do not expand our analysis on the concept of social role here. A detailed account of this concept including a specification and execution of an example role-assignment protocol will be presented in a separate paper.

2.3 Formalisation: Action Languages

We provide two alternative accounts of the presented theoretical framework using two different action formalisms. One account is formalised by means of the $\mathcal{C}+$ language while the other is formalised using EC. The $\mathcal{C}+$ language was chosen for the following reasons:

—It is a formalism with explicit transition systems semantics and provides support for the effects (direct and indirect) of actions and default persistence ('inertia') of facts from state to state.

—There exists a software tool, the Causal Calculator (CCALC), that supports a number of computational tasks regarding $\mathcal{C}+$ formalisations.

EC was chosen because it is a formal, yet intuitive action language. Like the $\mathcal{C}+$ language, it may represent actions with conditional and indirect effects, and the inertia of facts. An evaluation of the utility of EC and the $\mathcal{C}+$ language for the specification of open agent societies will be presented in a later section.

In this paper we follow two different routes to execute a society specification:

(1) We employ CCALC in order to execute the specifications formalised in the $\mathcal{C}+$ language.

(2) We develop the Society Visualiser (SV), a software tool that performs computational tasks on EC axiomatisations. SV executes the specifications formalised in EC.

We refer to these software tools as our *computational framework*. (CCALC is not the only means by which a $\mathcal{C}+$ formalisation may be executed, nor is SV the only means by which an EC axiomatisation may be executed.) Sections 3 and 4 review the $\mathcal{C}+$ language and CCALC. Sections 5 and 6 present a specification and execution of a variation of the Contract-Net Protocol (CNP) with the use of these technologies. Sections 7 and 8 review EC and the SV implementation, respectively, and Sections 9 and 10 present an EC specification and a SV execution of the CNP.

3. THE $\mathcal{C}+$ LANGUAGE

As already mentioned, $\mathcal{C}+$ is an action language with an explicit transition systems semantics. We describe here the version of $\mathcal{C}+$ presented in [Giunchiglia et al. 2004]. A detailed presentation placing more emphasis on the transition system semantics is given in [Sergot 2004a].

3.1 Basic Definitions

A *multi-valued propositional signature* is:

—a set σ of symbols called *constants*, and

—for each constant $c \in \sigma$, a non-empty finite set $dom(c)$ of symbols, disjoint from σ, called the *domain* of c.

For simplicity, in this presentation we will assume that every domain contains at least two elements.

An *atom* of signature σ is an expression of the form $c = u$ where $c \in \sigma$ and $u \in dom(c)$. A Boolean constant is one whose domain is the set of truth values $\{\mathsf{true}, \mathsf{false}\}$. When c is a Boolean constant we often write c for $c = \mathsf{true}$ and $\neg c$ for $c = \mathsf{false}$. A *formula* φ of signature σ is any propositional combination of atoms of σ. An *interpretation* I of σ is a function that maps every constant in σ to an element of its domain. An interpretation I *satisfies* an atom $c = u$ if $I(c) = u$. The satisfaction relation is extended from atoms to formulas according to the standard truth tables for the propositional connectives. A *model* of a set X of formulas of signature σ is an interpretation of σ that satisfies all formulas in X. If every model of a set X of formulas satisfies a formula φ then X *entails* φ, written $X \models \varphi$.

3.2 Syntax

The representation of an action domain in $\mathcal{C}+$ consists of *fluent* constants and *action* constants.

—Fluent constants are symbols characterising a state. They are divided into two categories: simple fluent constants and statically determined fluent constants. Simple fluent constants are related to actions by *dynamic laws* (that is, laws describing a transition from a state s_i to its successor state s_{i+1}). Statically determined fluent constants are characterised by *static laws* (that is, laws describing an individual state) relating them to other fluent constants. Static laws can also

be used to express constraints between simple fluents. Static and dynamic laws are defined below.

—Action constants are symbols characterising state transitions.

An *action signature* is a non-empty set σ^{f} of fluent constants and a non-empty set σ^{act} of action constants. An *action description* D in $\mathcal{C}+$ is a non-empty set of *causal laws* that define a transition system of a particular type. A causal law can be either a *static law* or a *dynamic law*. A static law is an expression[1]

$$F \text{ if } G \tag{1}$$

where F and G are formulas of fluent constants. In a static law, constants in F and G are evaluated on the same state. A dynamic law is an expression

$$F \text{ if } G \text{ after } H \tag{2}$$

where F, G and H are formulas such that every constant occurring in F is a simple fluent constant, every constant occurring in G is a fluent constant, and H is any combination of fluent constants and action constants. In a transition from state s_i to state s_{i+1}, constants in F and in G are evaluated on s_{i+1}, fluent constants in H are evaluated on s_i and action constants in H are evaluated on the transition itself. F is called the *head* of the static law (1) and the dynamic law (2).

The full $\mathcal{C}+$ language also provides *action dynamic laws*, which are expressions of the form

$$\alpha \text{ if } H$$

where α is a formula containing action constants only and H is a formula of action and fluent constants. We will not use action dynamic laws in this paper and so omit the details in the interests of brevity.

The $\mathcal{C}+$ language provides various abbreviations for common forms of causal laws. Those used in this paper are as follows:

—An expression of the form

$$\text{default } F$$

expresses that the formula F of fluent constants is assumed to hold by default in the absence of information to the contrary. It is an abbreviation for the static law:

$$F \text{ if } F$$

—A dynamic law of the form

$$\bot \text{ if } \top \text{ after } \alpha \wedge H$$

where α is a formula containing only action constants and H is a formula containing only fluent constants is abbreviated as:

$$\text{nonexecutable } \alpha \text{ if } H$$

[1]For brevity, we omit the keyword caused which appears at the beginning of static and dynamic laws in the original presentation of $\mathcal{C}+$ [Giunchiglia et al. 2004].

In the case where H is $\top$ then the abbreviation can be written as follows:

$$\mathsf{nonexecutable}\ \alpha$$

—The inertia of a fluent constant c over time is represented as:

$$\mathsf{inertial}\ c$$

This is an abbreviation for the *set* of dynamic laws of the form (for all values $u \in dom(c)$):

$$c = u \ \mathsf{if}\ c = u \ \mathsf{after}\ c = u$$

As already mentioned, a $\mathcal{C}+$ action description is a non-empty set of causal laws. Of particular interest is the sub-class of *definite* action descriptions. A $\mathcal{C}+$ action description D is *definite* if:

—the head of every causal law of D is an atom or $\bot$, and
—no atom is the head of infinitely many causal laws of D.

All the $\mathcal{C}+$ action descriptions in this paper will be definite.

3.3 Semantics

It is not possible in the space available here to give a full account of the $\mathcal{C}+$ language and its semantics. We trust that the $\mathcal{C}+$ language, and especially its abbreviations, are sufficiently natural that readers can follow the presentation of the case study in later sections. Interested readers are referred to [Giunchiglia et al. 2001; Giunchiglia et al. 2004] and [Sergot 2004a] for further technical details. For completeness, we summarise here the semantics of *definite* action descriptions as presented in [Sergot 2004a], ignoring (as we are) the presence of action dynamic laws (and assuming that the domain of every constant contains at least two elements).

Every action description D of $\mathcal{C}+$ defines a labelled transition system, as follows:

—States of the transition system are interpretations of the fluent constants σ^{f}. It is convenient to identify a state s with the set of fluent atoms satisfied by s (in other words, $s \models f = v$ iff $f = v \in s$ for every fluent constant f).
Let $T_{\mathrm{static}}(s)$ denote the heads of all static laws in D whose conditions are satisfied by s:

$$T_{\mathrm{static}}(s) =_{\mathrm{def}} \{F \mid \text{static law (1) is in } D, s \models G\}$$

For a definite action description D, an interpretation s of σ^{f} is a *state* of the transition system defined by D (or simply, a state of D for short) when

$$s = T_{\mathrm{static}}(s) \cup \mathit{Simple}(s)$$

where $\mathit{Simple}(s)$ denotes the set of simple fluent atoms satisfied by s. (So $s - \mathit{Simple}(s)$ is the set of statically determined fluent atoms satisfied by s.)

—Transition labels of the transition system defined by D (also referred to as *events* or *actions*) are the interpretations of the action constants σ^{act}.
A *transition* is a triple (s, ϵ, s') in which s is the initial state, s' is the resulting state, and ϵ is the transition label (or event). Since transition labels are interpretations of σ^{act}, it is meaningful to say that a transition label ϵ satisfies a formula α of σ^{act}: when $\epsilon \models \alpha$ we sometimes say that the transition (s, ϵ, s') is of type α.

—Let $E(s, \epsilon, s')$ denote the heads of all dynamic laws of D whose conditions are satisfied by the transition (s, ϵ, s'):

$$E(s, \epsilon, s') =_{\text{def}} \{F \mid \text{dynamic law (2) is in } D, s' \models G, s \cup \epsilon \models H\}$$

For a definite action description D, (s, ϵ, s') is a *transition of* D (or in full, a transition of the transition system defined by D) when s and s' are interpretations (set of atoms) of σ^{f} and ϵ is an interpretation of σ^{act} such that:
(1) $s = T_{\text{static}}(s) \cup \mathit{Simple}(s)$ (s is a state of D)
(2) $s' = T_{\text{static}}(s') \cup E(s, \epsilon, s')$

For any non-negative integer m, a *path* or *history* of D of length m is a sequence

$$s_0\, \epsilon_0\, s_1\, \ldots\, s_{m-1}\, \epsilon_{m-1}\, s_m$$

where $(s_0, \epsilon_0, s_1), \ldots, (s_{m-1}, \epsilon_{m-1}, s_m)$ are transitions of D.

4. THE CAUSAL CALCULATOR

The Causal Calculator (CCALC) is a system designed and implemented by the Action Group of the University of Texas for representing action and change in the $\mathcal{C}+$ language and performing a range of computational tasks on the resulting formalisations. CCALC has been applied to several 'challenge problems' (see, for example, [Akman et al. 2004; Lifschitz 2000; Lifschitz et al. 2000]).

CCALC has two inputs: a definite $\mathcal{C}+$ action description D and a query concerning D. We describe the *functionality* of CCALC, that is, the type of query that CCALC may compute, the *operation* of CCALC, that is, how the computation of a query is performed, and the way we use CCALC to execute an open agent society specification.

4.1 Functionality of the Causal Calculator

The functionality of CCALC includes computation of three kinds of tasks, each of which can be represented as a query:

—Prediction. Given (partial or complete) information about an initial state and a complete sequence of transitions, compute the information that holds in the resulting state(s) (if any) of a given transition system (action description) D.
—Planning. Given (partial or complete) information about an initial state and (partial or complete) information about a resulting state, compute the complete sequence(s) of transitions (if any) that will lead from the initial state to the resulting state of a given transition system (action description) D.
—Postdiction. Given, possibly, partial information about an initial state, (partial or complete) information about a resulting state, and, possibly, a (partial or complete) sequence of transitions that leads from the initial state to the resulting one, compute some additional information that holds in the initial state (if one exists) of a given transition system (action description) D.

In all of these computational tasks, information (partial or complete) about intermediate states (if any) may be provided. Apart from the information mentioned above, a query specifies the maximum number of transitions that a solution (that is, a computation of the query) may include.

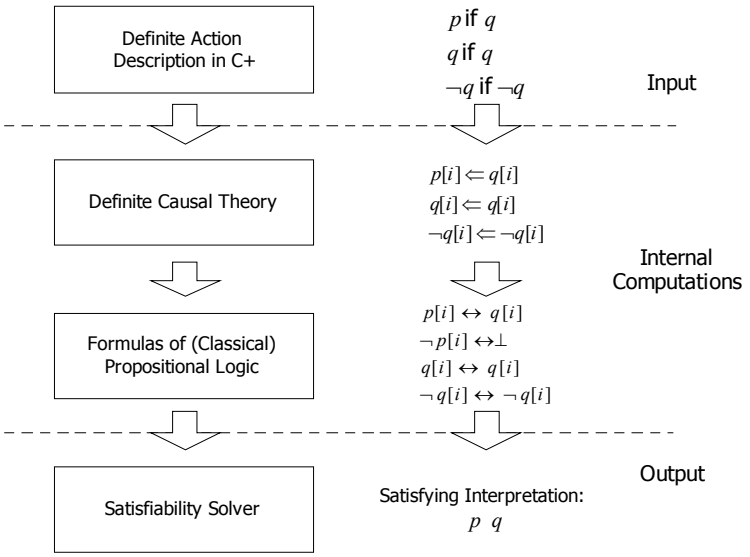

Fig. 1. The Operation of the Causal Calculator.

4.2 Operation of the Causal Calculator

Action descriptions in $\mathcal{C}+$ are translated by CCALC first into the language of *causal theories* [Giunchiglia et al. 2004] and then into propositional logic. The (ordinary, classical) models of the propositional theory correspond to paths in the transition system described by the original action description in $\mathcal{C}+$. In brief, CCALC performs the following tasks in order to compute a query. First, it translates a given definite action description D to a *definite causal theory* Γ_m^D. m specifies the length of paths to be considered. Second, it translates the definite causal theory Γ_m^D into a set of (ordinary, classical) propositional formulas $comp(\Gamma_m^D)$. Third, it invokes a satisfiability (SAT) solver [Kautz and Selman 1992] to find models of the propositional formulas $comp(\Gamma_m^D)$ which also satisfy the query. Figure 1 gives an overview of CCALC's operation.

We use CCALC in the following way: we specify the social constraints of an open agent society as a definite $\mathcal{C}+$ action description and use CCALC to evaluate queries about that action description. For present purposes it is sufficient to view CCALC as a 'black box' (see the dotted lines in Figure 1) without having to describe the internal computations. The reader is referred to [Giunchiglia et al. 2004] for a more detailed description of CCALC's operation.

4.3 Executing Open Agent Societies with the Causal Calculator

This section describes the way we use prediction, planning and postdiction queries to execute the specification of the social constraints of an open agent society. In each type of query, CCALC has as input a definite $\mathcal{C}+$ action description D^{soc} expressing the specification of social constraints. We refer to the states of the transition system defined by D^{soc} as *social states*. In other words, a social state is an interpretation

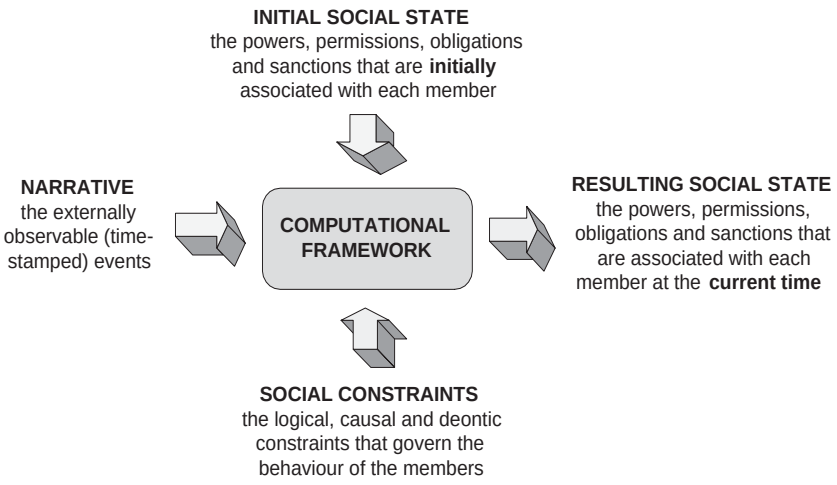

Fig. 2. Executing the Specification of an Open Agent Society: Prediction Queries.

(with some further properties) of the fluent constants of D^{soc}. These constants represent, amongst other things, the powers, permissions, obligations and sanctions of a society's members.

—Prediction queries. The computation of this type of query involves an initial social state, that is, a description of the powers, permissions, obligations and sanctions that are initially associated with the members of a society, and a *narrative*, that is, a description of temporally-sorted externally observable events of the society (a narrative is expressed as a sequence of transitions). The outcome of a prediction query (if any) is the current social state, that is, a description of the powers, permissions, obligations and sanctions that are associated with the members of the society at the current time (see Figure 2).

—Planning queries. Agents may compute planning queries: (i) at design-time in order to generate plans that will facilitate them in avoiding run-time conflicts (say), and (ii) at run-time in order to update their plans.

—Postdiction queries. New members of a society may seek to determine the past states of that society. Similar information may be requested by agents that have 'crashed' and resumed their operation. Such information can be produced via the computation of postdiction queries.

The computation of prediction, planning and postdiction queries may be additionally used to prove properties of the social constraints' specification; in Section 6 we prove properties of a specification by means of planning query computation.

5. SPECIFYING THE CNP IN THE $\mathcal{C}$+ LANGUAGE

In order to illustrate the use of the $\mathcal{C}$+ language and CCALC for specifying and executing open agent societies we present a $\mathcal{C}$+ specification and a CCALC execution of a well-studied protocol in the MAS field, the Contract-Net Protocol (CNP)

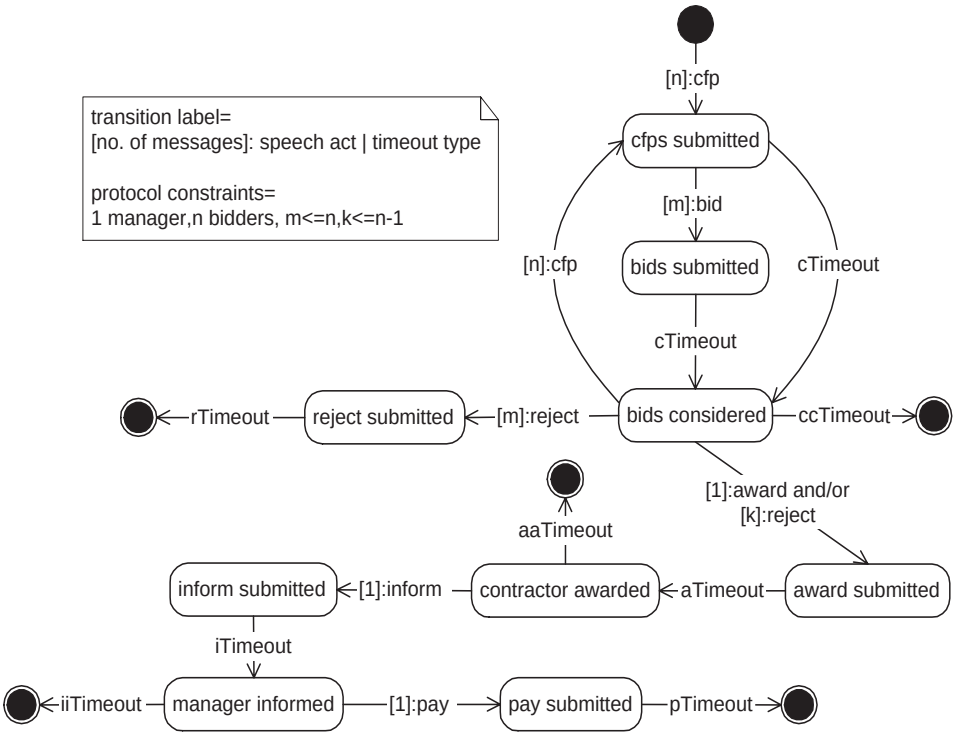

Fig. 3. The UML State Transition Diagram of the CNP.

[Smith and Davis 1978; Smith 1980; Davis and Smith 1983]. We focus on a variation of the CNP that is rich enough to illustrate the main strengths and weaknesses of the proposed executable specification of open agent societies.

5.1 The CNP

Our variation of the CNP is based on an abstract producer-consumer scenario [Pitt et al. 2001] where explorer agents (producers) sell information to cartographers (consumers). The information commodity in question is geophysical in nature, with explorer agents mapping out the distribution of oil in their environment. The geophysical data is of variable quality; it is in the explorers' interests to find the best possible oil 'plots', as these will fetch the best price on the market. It is in the cartographers' interests to get the best quality exploration data as these have the greatest intrinsic value. Trading is, of course, competitive, with cartographers initiating contract-net protocols for specific exploration.

Figure 3 shows a Unified Modelling Language (UML) state transition diagram of our variation of the CNP. A brief description of this protocol is the following: a cartographer (manager) issues a Call For Proposals (CFP) for a particular task (region exploration) to a set of explorers (bidders). Explorers submit their bids (if they are interested) to the cartographer. The cartographer then has three choices:

Table I. A Subset of the Action Signature of D^{CNP}.

Notation:
$Agent$, $Agent_2$, C, E range over $\{c_1, e_1, \ldots, e_n\}$
$Perf$ ranges over $\{cfp, bid, award, reject, inform, pay\}$
$Content$, $Content_2$ range over a finite set of task descriptions
$Round$ ranges over $\mathbb{Z}^+$, the set of positive integers
Rigid constants (domain is $\{cartographer, explorer\}$):
$role\_of(Agent)$
Simple fluent constants (Boolean domain):
$validActionHappened(Agent, Perf, Agent, Content, Round)$
Statically determined fluent constants (Boolean domain):
$pow(Agent, Perf, Agent, Content, Round)$,
$per(Agent, Perf, Agent, Content, Round)$,
$obl(Agent, Perf, Agent, Content, Round)$,
$sanctioned(Agent)$
Action constants (Boolean domain):
$valid(Agent, Perf, Agent, Content, Round)$

(i) award a particular bid (and reject the remaining bids), (ii) reject all received bids, or (iii) issue a new CFP incrementing the protocol round[2]. In the first case the awarded explorer should report the outcome of the awarded task and, provided that it does so, the cartographer should pay the awarded explorer. In the second case the protocol ends. In the third case the protocol starts again. Actions must be performed according to specified deadlines (timeouts).

5.2 Setting the Scene

This section presents an action description D^{CNP} that expresses the specification of the social constraints of the CNP. Table I shows a subset of the action signature $\sigma^{\mathrm{f}} \cup \sigma^{\mathrm{act}}$ of D^{CNP}. Variables are written with an upper-case first letter and constants start with a lower-case letter. Due to restrictions in CCalc's input language (our $\mathcal{C}+$ formalisation is translated to CCalc's input language in order to perform computational experiments), it is not possible to include nested constants of the form $pow(E, bid(E, C, Content, Round))$. We overcome this syntactical limitation by expressing such constants as $pow(E, bid, C, Content, Round)$.

For simplicity in this example CNP specification, we assume that agents do not change roles during the execution of the protocol. (We assume that agents have already been awarded roles via the execution of a role-assignment protocol — see Section 2.2.) Accordingly, the $role\_of(Agent)$ fluent constants are declared to be

[2]The protocol round is a parameter of the description of the agents' actions. It is represented as an integer that, initially, is equal to one and is incremented by one at the performance of the first valid CFP after a $cTimeout$ (see Figure 3).

'rigid' by means of $\mathcal{C}+$ causal laws:

$$\textsf{rigid } role\_of(Agent) \tag{3}$$

For a fluent constant f, the expression '$\textsf{rigid } f$' is a standard $\mathcal{C}+$ abbreviation for the set of dynamic causal laws [Giunchiglia et al. 2004]:

$$\bot \textsf{ if } \neg(f = v) \textsf{ after } f = v, \quad \text{for all } v \in dom(f)$$

There are some computational advantages in declaring fluent constants to be 'rigid'. Alternatively, we could have used non-rigid fluent constants to represent the agents' roles, thus allowing agents to occupy different roles during the execution of the protocol. Such a setting requires a further set of constraints governing the ways in which agents may change roles as the protocol progresses. Although such constraints can be expressed in the $\mathcal{C}+$ language, they would increase the number of causal laws of D^{CNP} and thereby compromise CCALC's efficiency in computing queries on the D^{CNP} action description.

Each (non-rigid) simple fluent constant of D^{CNP} is inertial. We express this constraint in $\mathcal{C}+$ as follows (for every non-rigid simple fluent constant c):

$$\textsf{inertial } c \tag{4}$$

We also add the following dynamic causal laws to specify that exactly one action takes place at each state transition:

$$\textsf{nonexecutable } \alpha_i \wedge \alpha_j \tag{5}$$

for all (Boolean) $\alpha_i, \alpha_j \in \sigma^{\text{act}}, \alpha_i \neq \alpha_j$, and

$$\textsf{nonexecutable } \neg\alpha_1 \wedge \neg\alpha_2 \wedge \ldots \wedge \neg\alpha_n \tag{6}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_n$ are the action constants of σ^{act} (all of which are Boolean in this example). The restriction expressed by (5) and (6) is not essential for the CNP specification but it is convenient when analysing runs (executions) of the protocol in later sections.

We now present a number of causal laws of D^{CNP} to specify, amongst other things, the powers, permissions, obligations and sanctions of the CNP participants.

5.3 Social Constraints

5.3.1 *Physical Capability.* This specification of the CNP includes *timeout events* (see Figure 3). These are *system events*, in the sense that they are performed by a global clock. When the CNP commences (this happens when the cartographer issues a valid CFP) a global clock starts 'ticking'. The timeout events are represented by a number of action constants: *cTimeout* is a timeout that takes place after a valid CFP, *aTimeout* is a timeout that takes place after a valid award, and so on. We introduce fluent constants to record that a timeout has taken place — for example, we use the *cTimeoutHappened* constants to record that a *cTimeout* has taken place and the *timeoutHappened* constants to record that some other timeout has taken place[3]. (Due to space limitations, several action and fluent constants of

[3]We are aware that encoding fluent constants representing the history of the protocol in a state description runs counter to the spirit of transition systems. We discuss this issue in Section 11.

the action signature of D^{CNP}, including the ones mentioned above, are not presented in Table I.) The effects of a timeout are expressed in terms of the states of affairs it initiates and terminates. An *aTimeout*, for example, obliges the awarded explorer to report the result of the awarded task.

The remaining actions of the CNP specification are those performed by the participants of the protocol (that is, CFP, bid, award, reject, inform and pay). We have chosen to specify that every agent is capable of performing any of these actions at any time. For example, an agent is always capable of broadcasting a CFP. In order to express the effects of this action, we need to distinguish between the act of ('successfully') issuing a CFP and the act by means of which that CFP is issued. (Similar distinctions need to be made for expressing the effects of the remaining actions that agents may perform.) Communicating a CFP for a task t, by means of broadcasting a message of a particular form via a TCP/IP socket connection, for example, is not necessarily valid ('successful'), in the sense that it empowers the recipients to bid for t. It is only if the CFP is communicated by an agent with the institutional power to issue the CFP that it will be valid (see Section 2.1.1). A specification of institutional power and valid actions in the context of the CNP is presented next.

5.3.2 *Institutionalised Power and Valid Actions.* Institutional powers are represented with the use of the statically determined fluent constants *pow*. There are several laws that define ('statically determine') the *pow* fluents. For example, the power to bid is defined as follows:

$$\begin{aligned} &pow(E, bid, C, RelatedContent, Round) \text{ if} \\ &\quad validActionHappened(C, cfp, E, Content, Round) \wedge \\ &\quad matches(Content, RelatedContent) \wedge \\ &\quad \neg cTimeoutHappened(Round) \wedge \\ &\quad \neg validBidIssued(E, Round) \end{aligned} \tag{7}$$

The above law specifies that an explorer E has the power to bid for a task in a protocol round if the following conditions hold:

—The cartographer C has issued a valid CFP in that protocol round. (We record valid actions with the use of the *validActionHappened* simple fluent constants.)

—The task that the explorer is empowered to bid for is 'related' (similar) to the task described in the valid CFP. (The *matches* (rigid) constants specify whether two tasks are 'related' or not and are defined in an application-specific manner.)

—A *cTimeout* has not taken place in that protocol round.

—E has not issued any valid bids in that protocol round. (A *validBidIssued* simple fluent constant expresses whether an explorer has issued a valid bid in a protocol round without indicating the task described in the bid.)

Only when these conditions hold will an explorer be empowered to bid. The 'closed-world assumption' regarding the specification of powers is expressed as follows:

$$\text{default } \neg pow(Agent, Perf, Agent_2, Content, Round) \tag{8}$$

In other words, in the absence of information to the contrary (for instance, constraint (7)), no agent is empowered to perform an action. The specification of

the power to issue a CFP, award, reject, inform and pay is expressed in a similar manner.

Valid actions are determined in terms of institutional powers. In this example CNP specification, we express valid actions as follows:

$$\begin{aligned}&\mathsf{nonexecutable}\ \mathit{valid}(\mathit{Agent}, \mathit{Perf}, \mathit{Agent}_2, \mathit{Content}, \mathit{Round})\ \mathsf{if}\\ &\quad \neg \mathit{pow}(\mathit{Agent}, \mathit{Perf}, \mathit{Agent}_2, \mathit{Content}, \mathit{Round})\end{aligned} \tag{9}$$

The above constraint specifies that it is not possible for an action to be valid if the agent that performed it did not have the institutional power to do so. Constraint (9) is but one of the possible formalisations of valid actions. (See, for example, how powers and valid actions are formalised in [Sergot 2004a].)

It is now possible to express the effects of an action, given that we may determine whether or not it is valid. An explorer E's valid bid for a task t, for instance, empowers the cartographer to award t to E (after the occurrence of a timeout). An invalid bid does not have any effects on the cartographer's powers. In this example CNP specification, we have chosen to ignore invalid actions (and, therefore, σ^{act} does not include constants expressing such actions).

Every transition of the transition system defined by D^{CNP} is labelled either with a valid action or a timeout event, due to the fact that exactly one action takes place at each transition (constraints (5) and (6)) and invalid actions are ignored. The number of transitions l of the longest path of the transition system defined by D^{CNP} can be determined from the number of agents n occupying the role of the explorer and the maximum number r of protocol rounds:

$$\begin{aligned} l &= (\overbrace{n}^{cfp} + \overbrace{n}^{bid} + \overbrace{1}^{cTimeout}) \times r + \overbrace{1}^{award} + \overbrace{n-1}^{reject} \\ &\quad + \overbrace{1}^{aTimeout} + \overbrace{1}^{inform} + \overbrace{1}^{iTimeout} + \overbrace{1}^{pay} + \overbrace{1}^{pTimeout} \\ &= (2 \times n \times r) + r + n + 5 \end{aligned} \tag{10}$$

Intuitively, the longest path includes the following sequence of events: the cartographer issues CFPs to all explorers, all explorers bid and the first timeout takes place. This sequence of events is repeated r times, that is, the maximum number of protocol rounds. Then, the cartographer awards an explorer and rejects all remaining ones. The awarded explorer reports the result of the awarded task and the cartographer pays the explorer. This sequence of actions includes all timeouts that specify the interval during which the aforementioned actions may be performed.

5.3.3 *Permission.* We now specify what actions, valid or invalid, are to be classified as permitted or obligatory. Like the *pow* fluent constants, we represent permitted and obligatory actions with the use of statically determined fluent constants, the *per* and *obl* constants. For this CNP specification, we have chosen to say that an agent is permitted to perform an action if and only if it has the institutional power to perform that action:

$$\begin{aligned}&\mathit{per}(\mathit{Agent}, \mathit{Perf}, \mathit{Agent}_2, \mathit{Content}, \mathit{Round})\ \mathsf{if}\\ &\quad \mathit{pow}(\mathit{Agent}, \mathit{Perf}, \mathit{Agent}_2, \mathit{Content}, \mathit{Round})\end{aligned} \tag{11}$$

$$\mathsf{default}\ \neg \mathit{per}(\mathit{Agent}, \mathit{Perf}, \mathit{Agent}_2, \mathit{Content}, \mathit{Round}) \tag{12}$$

According to constraints (11) and (12), a participant of the CNP is always permitted to exercise its institutional powers. As already mentioned in Section 2.1.1, however, there is no fixed relationship between permission and power. We could have specified, for example, that the cartographer is never permitted to broadcast more than one CFP, although sometimes empowered to do so, or an explorer is always permitted to bid, although not always empowered to do so. In other words, an action may be: (i) valid and permitted, (ii) valid and forbidden, (iii) invalid and permitted, or (iv) invalid and forbidden. For simplicity, in this example an action may be either valid and permitted or invalid and forbidden. (See [Artikis et al. 2003a] for an example protocol specification in which permission does not coincide with institutional power.)

Obligations arise in three situations, one of which is expressed as follows:

$$\begin{aligned} &obl(E, inform, C, Result, Round) \text{ if} \\ &\quad validActionHappened(C, award, E, Content, Round) \wedge \\ &\quad result\_of(Content) = Result \wedge \\ &\quad timeoutHappened(a) \wedge \\ &\quad \neg timeoutHappened(aa) \wedge \\ &\quad \neg validActionHappened(C, inform, E, Result, Round) \end{aligned} \tag{13}$$

Constraint (13) states that issuing a valid award to an explorer obliges that explorer, after an *aTimeout* occurs, to report the result of the awarded task. (The *result_of* constants, declared to be 'rigid', specify the outcome of the completion of the awarded task and are defined in an application-specific manner.) The explorer will discharge this obligation if it reports the result of the awarded task by the specified deadline (*aaTimeout*). In more realistic scenarios, reporting the result of the awarded task would not be enough to discharge this obligation; it should also be externally observable that the awarded task has been completed. In order to simplify our presentation we do not accommodate such complications.

As in the case of powers and permissions, the 'closed-world assumption' regarding obligations is specified as follows:

$$\text{default } \neg obl(Agent, Perf, Agent_2, Content, Round) \tag{14}$$

The above definitions of permitted and obligatory actions express that performing actions when not empowered to do so, and not reporting the result of the awarded task in the given interval, is considered 'unacceptable' behaviour.

5.3.4 *Enforcement Policies.* In the present example, sanctions deal with a particular type of 'unacceptable' behaviour, that which results from non-compliance with obligations. For instance, the awarded contractor's failure to report the result of the awarded task by the specified time (*aaTimeout*) creates a sanction for that contractor:

$$\begin{aligned} &sanctioned(E) \text{ if} \\ &\quad timeoutHappened(aa) \wedge \\ &\quad current\_round = Round \wedge \\ &\quad validActionHappened(C, award, E, Content, Round) \end{aligned} \tag{15}$$

The simple fluent constant *current_round* represents the current protocol round.

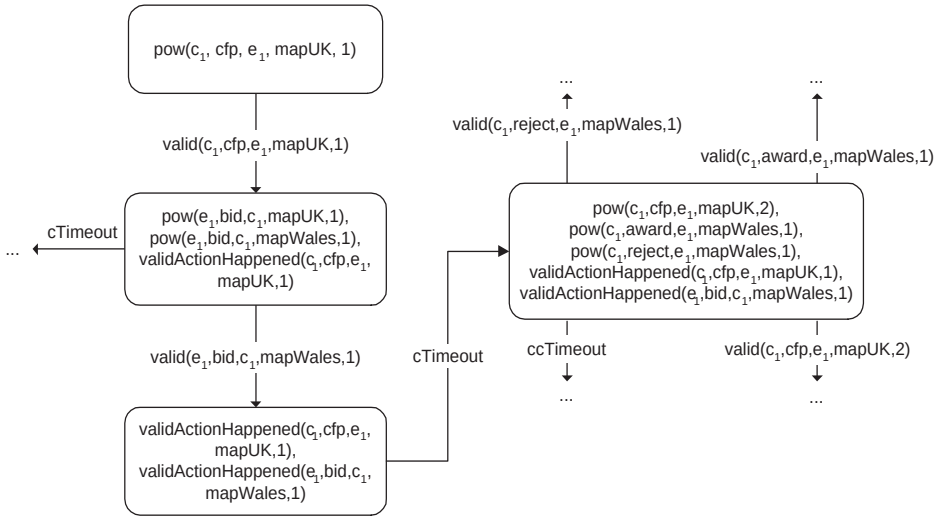

Fig. 4. A Fragment of the Labelled Transition System Defined by D^{CNP} (Two Participants).

Similarly, the end-state that is reached when a timeout (that is, *iTimeout*) occurs at the 'manager informed' state (see Figure 3) includes a sanction for the manager. In this protocol specification, sanctions are just recorded, that is, it is just pointed out that some agent has exhibited 'unacceptable' or 'anti-social' behaviour. There is no explicit penalty associated with a sanction. There are many other possible specifications of permitted and obligatory actions, and enforcement policies. The one presented is chosen for the sake of providing a concrete illustration.

5.4 Social States

Figure 4 presents a fragment of the labelled transition system defined by D^{CNP} with two participants: cartographer c_1 and explorer e_1. Due to constraints (5) and (6), each edge of the diagram is labelled with exactly one action constant. For clarity, Figure 4 shows only the *pow* and *validActionHappened* fluent constants at each state. There are no loops in the transition system defined by D^{CNP} because we record the valid actions and timeouts that have taken place — therefore, each state is cumulative with respect to the *validActionHappened*, *timeoutHappened* and other fluent constants recording the history of the protocol.

In the initial state of the protocol, c_1 is empowered to issue a CFP about a task, say *mapUK*. Following the valid CFP of c_1, e_1 is empowered to bid for *mapUK* as well as a 'related' task, described as *mapWales*. Similarly, the following states express the powers of the CNP participants according to the D^{CNP} laws. Note that the transition system presented in Figure 4 is at a different level of abstraction from the UML state transition diagram presented in Figure 3. Figure 4 presents a detailed account of the transition system, for instance, showing a subset of an interpretation of σ^{f} at each of its states whereas the UML diagram simply associates an informal textual description with each of its states.

We may prove various properties of the transition system defined by D^{CNP} (with

two or more participants). Consider the following examples.

PROPOSITION 1. *The transition system defined by D^{CNP} does not include a state in which an agent is empowered, but not permitted, to perform an action.*

PROOF. Assume that

$$s \supseteq \{pow(Agent, Perf, Agent_2, Content, Round), \\ \neg per(Agent, Perf, Agent_2, Content, Round)\}$$

is a state of the transition system defined by D^{CNP}. Since s is a state of D^{CNP}, then it is an interpretation of σ^{f} that satisfies '$G \rightarrow F$' for every static law of the form 'F if G' in D^{CNP}. ($\rightarrow$ is material implication.) However, D^{CNP} includes static law (11) and s does not satisfy

$$pow(Agent, Perf, Agent_2, Content, Round) \rightarrow \\ per(Agent, Perf, Agent_2, Content, Round)$$

Therefore, s is not a state of D^{CNP}. □

PROPOSITION 2. *The transition system defined by D^{CNP} does not include a state in which an agent is obliged to report the result of a task in a protocol round without already being awarded that task in that round.*

PROOF. Assume that

$$s \supseteq \{\neg validActionHappened(C, award, E, Content, Round), \\ obl(E, inform, C, result\_of(Content), Round)\}$$

is a state of D^{CNP}. Given static law (14), the only law that makes true a constant of the form $obl(E, inform, C, result\_of(Content), Round)$ is static law (13). This law requires, amongst other conditions, that the following fluent is satisfied: $validActionHappened(C, award, E, Content, Round)$. However, this is not consistent with our initial assumption, that is

$$s \supseteq \{\neg validActionHappened(C, award, E, Content, Round)\}$$

Therefore, s is not a state of D^{CNP}. □

6. EXECUTING THE CNP WITH THE CAUSAL CALCULATOR

In order to perform computational experiments, we expressed our $\mathcal{C}+$ action description D^{CNP} in CCALC's input language [Lee et al. 2001; Akman et al. 2004]. We now present a number of queries submitted to CCALC and the results obtained. The tested version of the CNP has four participants, c_1 occupying the role of the cartographer, and e_1, e_2 and e_3 occupying the role of explorer. There is one task, described as $mapUK$, and the protocol can have at most two rounds.

According to formula (10), the longest path of the transition system defined by D^{CNP}, with three explorers and two protocol rounds, has twenty-two transitions:

$$l = (2 \times n \times r) + r + n + 5 = (2 \times 3 \times 2) + 2 + 3 + 5 = 22$$

Query 1 *(Prediction).* We are in a state where the following events have taken place: c_1 has issued valid CFPs to the three explorers about $mapUK$ in the first protocol round. All of the explorers have issued valid bids about the same task.

Finally the first timeout has elapsed. In this state, may c_1 issue a valid reject to e_1 regarding $mapUK$ in the first protocol round? If yes, what are the new powers associated with c_1?

CCALC determines that the $valid(c_1, reject, e_1, mapUK, 1)$ action is 'executable' (see constraint (9)). Moreover, as a result of this action, c_1 has the following powers:

$$pow(c_1, award, e_2, mapUK, 1),$$
$$pow(c_1, award, e_3, mapUK, 1),$$
$$pow(c_1, reject, e_2, mapUK, 1),$$
$$pow(c_1, reject, e_3, mapUK, 1).$$

Intuitively, in the resulting state the cartographer has no powers regarding e_1 because it rejected e_1's bid. However, the cartographer is still empowered to award or reject the remaining valid bids. Finally, the cartographer's rejection terminated its power to issue CFPs.

Query 2 *(Planning).* Given the initial state of the CNP, that is, one in which c_1 is empowered to issue a CFP to the three explorers in the first protocol round, is it possible to find a state, within twenty-two transitions, where some agent is empowered, but not permitted, to perform an action?

CCALC finds no solution within twenty-two transitions. In other words, a solution could not be found even in the longest path of the transition system defined by D^{CNP}. Therefore, there is *no* state in this protocol where an agent is empowered, but not permitted, to perform an action. The result of this query is consistent with Proposition 1.

Query 3 *(Planning).* Given the initial state of the CNP, find *all possible paths* (if any), within three transitions, that end in a state where c_1 is permitted but not obliged to reject an explorer's bid.

CCALC finds three solutions. They are the following:

```
Solution 1:
ACTIONS:  valid(c1, cfp, e3, mapUK, 1)
ACTIONS:  valid(e3, bid, c1, mapUK, 1)
ACTIONS:  cTimeout

Solution 2:
ACTIONS:  valid(c1, cfp, e1, mapUK, 1)
ACTIONS:  valid(e1, bid, c1, mapUK, 1)
ACTIONS:  cTimeout

Solution 3:
ACTIONS:  valid(c1, cfp, e2, mapUK, 1)
ACTIONS:  valid(e2, bid, c1, mapUK, 1)
ACTIONS:  cTimeout
```

According to D^{CNP}, the cartographer would be empowered, and therefore permitted, to reject a valid bid if it has not already rejected or awarded that bid. (Recall that, due to constraints (11) and (12), in this CNP specification permission coincides with power.) Moreover, the cartographer would be obliged to reject a valid bid if it has not already rejected or awarded this bid, and has awarded some other

valid bid. Like every action, a rejection should be performed in a particular interval (see Figure 3).

Query 4 *(Planning).* Given the initial state of the CNP, is it possible to find a state, within twenty-two transitions, where an agent is obliged to report the result of a task in a protocol round without already being awarded that task in that round?

CCALC finds no solution within twenty-two transitions. Given that the longest path of the transition system defined by D^{CNP} has twenty-two transitions, we infer that there is *no* state in the CNP where an agent is obliged to report the result of a task in a protocol round without already being awarded that task in that round. The result of this query is consistent with Proposition 2.

Note that a failure to find a solution to Query 4 within eighteen transitions (rather than twenty-two transitions) would be sufficient to conclude that the result of this query is consistent with Proposition 2. The maximum number of transitions that could lead to a state where an agent would be obliged to report the result of the awarded task is eighteen:

$$\begin{aligned} l &= (\overbrace{n}^{cfp} + \overbrace{n}^{bid} + \overbrace{1}^{cTimeout}) \times r + \overbrace{1}^{award} + \overbrace{n-1}^{reject} + \overbrace{1}^{aTimeout} \\ &= (2 \times n \times r) + r + n + 1 \\ &= (2 \times 3 \times 2) + 2 + 3 + 1 = 18 \end{aligned}$$

Recall that n represents the number of explorers and r the maximum number of protocol rounds.

Query 5 *(Postdiction).* Initially, c_1 was permitted to issue a CFP to e_3 regarding *mapUK* in the first protocol round. After one transition c_1 is empowered to award e_1 regarding *mapUK* in the first protocol round. What can we say about the powers of c_1 in the initial state?

CCALC finds several solutions. In all of them, the initial state includes the following:

$$\begin{aligned} &per(c_1, cfp, e_3, mapUK, 1), \\ &validActionHappened(e_1, bid, c_1, mapUK, 1), \\ &pow(c_1, cfp, e_3, mapUK, 1). \end{aligned}$$

At each solution, the action that emanates from the initial state is *cTimeout*, that is, the timeout after a valid CFP. At the initial state of all solutions neither e_1 nor e_3 is empowered to bid. e_1 has already issued a valid bid (therefore the last condition of constraint (7) is not satisfied) whereas e_3 has not received a valid CFP from the cartographer (therefore the first condition of constraint (7) is not satisfied). In some solutions e_2 is initially empowered to bid while in other solutions it does not have that power. At the final state in each solution no agent is empowered to bid because a *cTimeout* has taken place (consequently the third condition of constraint (7) is not satisfied).

7. THE EVENT CALCULUS

Now we discuss the use of an alternative action formalism, the Event Calculus (EC), for expressing the specifications of social constraints. EC [Kowalski and

Table II. Main Predicates of the Event Calculus.

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

Sergot 1986] is a formalism for representing and reasoning about actions or events and their effects in a logic programming framework. It is based on a many-sorted first-order predicate calculus. For the version used here, the underlying time model is linear and may include real numbers or integers. To make the account self-contained, this section describes the dialect of EC that we employ.

Where F is a *fluent* (a property that is allowed to have different values at different points in time) the term $F = V$ denotes that fluent F has value V. Boolean fluents are a special case in which the possible values are true and false. Informally, $F = V$ holds at a particular time-point if $F = V$ has been *initiated* by an event at some earlier time-point, and not *terminated* by another event in the meantime.

An *action description* in EC includes axioms that define, amongst other things, the action occurrences (with the use of happens predicates), the effects of actions (with the use of initiates and terminates predicates), and the values of the fluents (with the use of initially and holdsAt predicates). Table II summarises the main EC predicates. We maintain the convention of writing variables with an upper-case first letter, and predicates and constants with a lower-case first letter. The domain-independent definitions of the EC predicates are presented in the following section.

8. THE SOCIETY VISUALISER

In order to execute EC specifications of social constraints, we have developed a software tool, called the Society Visualiser (SV), that computes prediction queries regarding EC action descriptions. The computation of a prediction query involves a narrative (expressed by means of happens predicates), the specification of social constraints (expressed by means of initiates, terminates and holdsAt predicates) and an initial social state (an interpretation of the fluents of the action description in question expressed by means of initially or holdsAt predicates). The outcome of the query is the social state that is produced by the events described in the narrative (see Figure 2).

SV expresses an EC action description as a logic program. The domain-independent

definition of the holdsAt predicate is as follows:

$$\begin{aligned} &\mathsf{holdsAt}(F=V,T) \leftarrow \\ &\quad \mathsf{initially}(F=V), \\ &\quad \mathsf{not\ broken}(F=V,0,T) \end{aligned} \tag{16}$$

$$\begin{aligned} &\mathsf{holdsAt}(F=V,T) \leftarrow \\ &\quad \mathsf{happens}(Act,T'), \\ &\quad T' < T, \\ &\quad \mathsf{initiates}(Act,F=V,T'), \\ &\quad \mathsf{not\ broken}(F=V,T',T) \end{aligned} \tag{17}$$

According to axiom (16) 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 (17) 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* [Clark 1978]. The domain-independent predicate broken is defined as follows:

$$\begin{aligned} &\mathsf{broken}(F=V,T_1,T_3) \leftarrow \\ &\quad \mathsf{happens}(Act,T_2), \\ &\quad T_1 \leq T_2,\ T_2 < T_3, \\ &\quad \mathsf{terminates}(Act,F=V,T_2) \end{aligned} \tag{18}$$

$F=V$ 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:

$$\begin{aligned} &\mathsf{terminates}(Act,F=V,T) \leftarrow \\ &\quad \mathsf{initiates}(Act,F=V',T), \\ &\quad V \neq V' \end{aligned} \tag{19}$$

Axiom (19) 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 the 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 (16)–(19), specify that a fluent does not hold at the time that was initiated but holds at the time it was terminated. Second, in addition to their domain-independent definitions, the holdsAt and terminates predicates may be defined in a domain-dependent manner (the happens, initially and initiates predicates are defined only in a domain-dependent manner).

The social state of an open agent society is available to the agent designers and society designers via a Graphical User Interface (GUI). Figure 5 shows the GUI of SV during the simulation of an open agent society executing a contract-net protocol. The GUI displays the social roles, institutional powers, permissions, obligations, sanctions and valid actions that are associated with each agent and each institution (if more than one) that is part of a society. In addition, the interface displays information about the environment. The state of the environment holds information that is not presented in the display of the agents or the institutions — in

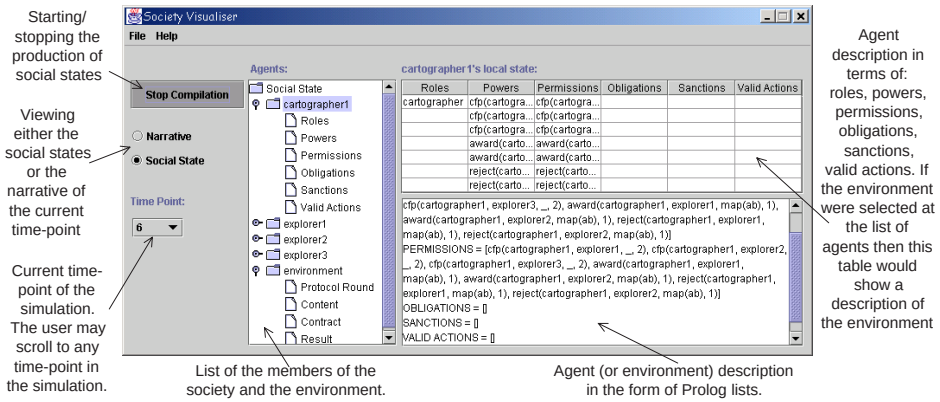

Fig. 5. The GUI of the Society Visualiser.

the example of a contract-net protocol, the environment may hold information such as the description of the task under negotiation (described as 'content' in Figure 5), the result of the completion of the awarded task, and other protocol-specific information.

9. SPECIFYING THE CNP IN THE EVENT CALCULUS

We demonstrate the use of EC and SV in specifying and executing open agent societies by presenting an EC specification and a SV execution of the CNP. Table III shows the main fluents and actions of the EC CNP specification. The syntax of the fluents of the EC specification differs from the syntax of the fluent constants of the $\mathcal{C}+$ specification (of the CNP) in that it is possible to express nested fluents in a logic program. (Recall that, due to restrictions in CCALC's input language, we had to modify the syntax of the nested fluent constants.) The style of the EC specification is also different from that of the $\mathcal{C}+$ specification in that the EC specification treats *pow*, *obl* and *sanctioned* as simple fluent constants (in the terminology adopted in the $\mathcal{C}+$ language) rather than statically determined fluent constants (as in the $\mathcal{C}+$ specification of the CNP). We employed this alternative treatment in order to illustrate the possible ways of specifying the constraints of an open agent society.

The EC action description expressing the CNP specification includes axioms (16)–(19). As a result, every fluent of this action description is inertial. The following section focuses on the specification of social constraints — we discuss the social states of the CNP in Section 10 when animating a run of this protocol.

9.1 Social Constraints

All communicative actions of the CNP (that is, CFP, bid, award, reject, inform and pay) may be performed at any time by any participant. The effects of a communicative action depend on whether or not that action is valid. An account of valid actions in the context of the EC CNP specification is presented next.

Table III. A Subset of the Fluents and Actions of the EC CNP Specification.

Fluents (Boolean domain):
$pow(Agent, Act)$, $per(Agent, Act)$, $obl(Agent, Act)$, $sanctioned(Agent)$
Actions (Boolean domain):
$cfp(Agent, Agent_2, Content, Round)$, $bid(Agent, Agent_2, Content, Round)$, $award(Agent, Agent_2, Content, Round)$, $reject(Agent, Agent_2, Content, Round)$, $inform(Agent, Agent_2, Result, Round)$, $pay(Agent, Agent_2, Result, Round)$

9.1.1 *Institutionalised Power and Valid Actions.* In the interests of brevity, for the purposes of this example we have chosen not to include an action labelled 'valid' in the EC specification — valid actions are represented implicitly whereas institutional powers are represented explicitly.

Issuing a valid CFP to a set of explorers empowers those explorers to bid:

$$\begin{aligned} &\mathsf{initiates}(Cfp, pow(E, Bid) = \mathsf{true}, T) \leftarrow \\ &\quad Cfp = cfp(C, E, Content, Round), \\ &\quad Bid = bid(E, C, RelContent, Round), \\ &\quad \mathsf{holdsAt}(pow(C, Cfp) = \mathsf{true}, T), \\ &\quad matches(Content, RelContent) \end{aligned} \tag{20}$$

Note that *matches* is treated as an ordinary predicate and not as a time-varying fluent. If the condition presented in the penultimate line of constraint (20) is not satisfied, then the CFP will be invalid and it will have no effects on the recipients' powers.

The power to bid is terminated by issuing a valid bid:

$$\begin{aligned} &\mathsf{initiates}(Bid, pow(E, Bid) = \mathsf{false}, T) \leftarrow \\ &\quad Bid = bid(E, C, Content, Round), \\ &\quad \mathsf{holdsAt}(pow(E, Bid) = \mathsf{true}, T) \end{aligned} \tag{21}$$

A valid bid may be performed by a particular deadline, that is, by the time a $cTimeout$ takes place:

$$\begin{aligned} &\mathsf{initiates}(cTimeout, pow(E, Bid) = \mathsf{false}, T) \leftarrow \\ &\quad Bid = bid(E, C, Content, Round), \\ &\quad \mathsf{holdsAt}(pow(E, Bid) = \mathsf{true}, T) \end{aligned} \tag{22}$$

All the powers in the CNP are propagated in the manner outlined above.

9.1.2 *Permission.* In this example, an agent is permitted to perform an action if and only if it has the power to perform that action:

$$\begin{aligned} &\mathsf{holdsAt}(per(Agent, Act) = \mathsf{true}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(pow(Agent, Act) = \mathsf{true}, T) \end{aligned} \tag{23}$$

$$\begin{aligned} &\mathsf{holdsAt}(per(Agent, Act) = \mathsf{false}, T) \leftarrow \\ &\quad \mathsf{not\ holdsAt}(per(Agent, Act) = \mathsf{true}, T) \end{aligned} \tag{24}$$

Obligations arise in three situations. For example, a timeout after a valid award ($aTimeout$) obliges the awarded explorer to report the result of the awarded task:

$$\begin{aligned}&\mathsf{initiates}(aTimeout, obl(E, Inform) = \mathsf{true}, T) \leftarrow \\ &\quad Inform = inform(E, C, Result, Round), \\ &\quad Award = award(C, E, Content, Round), \\ &\quad \mathsf{holdsAt}(validActionHappened(Award) = \mathsf{true}, T), \\ &\quad result\_of(Content, Result)\end{aligned} \tag{25}$$

Here *result_of* is an ordinary predicate and not a fluent of EC. The obligation mentioned above must be fulfilled by a particular deadline, that is, by the time an $aaTimeout$ takes place:

$$\begin{aligned}&\mathsf{initiates}(aaTimeout, obl(E, Inform) = \mathsf{false}, T) \leftarrow \\ &\quad Inform = inform(E, C, Result, Round), \\ &\quad \mathsf{holdsAt}(obl(E, Inform) = \mathsf{true}, T)\end{aligned} \tag{26}$$

The two remaining obligations are specified in a similar manner.

9.1.3 *Enforcement Policies.* In this CNP specification, sanctions arise when agents do not comply with their obligations. For example, we state the following regarding the obligation to report the outcome of the awarded task:

$$\begin{aligned}&\mathsf{initiates}(aaTimeout, sanctioned(E) = \mathsf{true}, T) \leftarrow \\ &\quad Inform = inform(E, C, Result, Round), \\ &\quad \mathsf{holdsAt}(obl(E, Inform) = \mathsf{true}, T)\end{aligned} \tag{27}$$

Recall that a fluent holds at the time that it was terminated, although it does not hold at the time that it was initiated (see Section 8). Therefore, even though the obligation to report the outcome of the task is terminated by the $aaTimeout$ (see constraint (26)), it still holds at time of the occurrence of this timeout. Thus, the condition of constraint (27) will hold (if an obligation was earlier created) and a sanction will be initiated. The two remaining sanctions are specified in a similar manner.

10. EXECUTING THE CNP WITH THE SOCIETY VISUALISER

Given a set of temporally-ordered events, we may determine (by issuing prediction queries to SV) the social states of the CNP. We present here an example run (execution) of the CNP by a group of four agents; three agents e_1, e_2 and e_3 occupying the role of the explorer, and an agent c_1 occupying the role of the cartographer. The run proceeds as follows (we assume that the model of time includes the set of positive integers): c_1 issues a CFP to the three explorers and then e_2 submits a bid. After the first timeout, e_1 submits a bid and then the cartographer issues a new CFP (having modified the task description and incremented the protocol round). Following the second CFP, e_2 submits its bid. The cartographer awards the bid of e_2. However, e_2 fails to report the result of the awarded task by the specified deadline.

We query SV in order to determine the social states of the CNP at each point in time. Consider, for instance, the following query, similar to Query 1 (Section 6) executed by CCALC on the $\mathcal{C}+$ action description D^{CNP}:

Table IV. Execution of the Specification of the CNP (Narrative and Cartographer).

Time	**Narrative**	**agent** c_1
0		$pow(c_1, cfp(c_1, e_1, Content, 1))$, $pow(c_1, cfp(c_1, e_2, Content, 1))$, $pow(c_1, cfp(c_1, e_3, Content, 1))$
1	$cfp(c_1, e_1, map(x), 1)$, $cfp(c_1, e_2, map(x), 1)$, $cfp(c_1, e_3, map(x), 1)$	$pow(c_1, cfp(c_1, e_1, Content, 1))$, $pow(c_1, cfp(c_1, e_2, Content, 1))$, $pow(c_1, cfp(c_1, e_3, Content, 1))$
2	$bid(e_2, c_1, map(w), 1)$	
3	$cTimeout$	
4	$bid(e_1, c_1, map(x), 2)$	$pow(c_1, award(c_1, e_2, map(w), 1))$, $pow(c_1, reject(c_1, e_2, map(w), 1))$, $pow(c_1, cfp(c_1, e_1, NewContent, 2))$, $pow(c_1, cfp(c_1, e_2, NewContent, 2))$, $pow(c_1, cfp(c_1, e_3, NewContent, 2))$
5	$cfp(c_1, e_1, map(y), 2)$ $cfp(c_1, e_2, map(y), 2)$ $cfp(c_1, e_3, map(y), 2)$	$pow(c_1, award(c_1, e_2, map(w), 1))$, $pow(c_1, reject(c_1, e_2, map(w), 1))$, $pow(c_1, cfp(c_1, e_1, NewContent, 2))$, $pow(c_1, cfp(c_1, e_2, NewContent, 2))$, $pow(c_1, cfp(c_1, e_3, NewContent, 2))$
6	$bid(e_2, c_1, map(z), 2)$	
7	$cTimeout$	
8	$award(c_1, e_2, map(z), 2)$	$pow(c_1, award(c_1, e_2, map(z), 2))$, $pow(c_1, reject(c_1, e_2, map(z), 2))$
9	$aTimeout$	
10	$aaTimeout$	

Query 6 *(Prediction).* We are in a state where the following events have taken place: c_1 has issued valid CFPs to the three explorers in the second protocol round. e_2 has issued a valid bid about a ('related') task in the same protocol round. Finally, the second timeout has elapsed. In this state, is c_1 empowered to award the task to e_2? If it is and c_1 does award the task to e_2, what are the new powers associated with c_1?

SV determines that c_1 is empowered to award the task to e_2. Moreover, after a valid award, c_1 has no institutional powers.

Tables IV and V present the results of a series of prediction queries submitted to SV. Query 6, for example, corresponds to time-point $T = 8$ of Table IV. Given a description of a run of the CNP in terms of the happens predicates, the institutional powers of each participant ag at each time-point t are generated by means of the following query:

$$? - \mathsf{holdsAt}(pow(ag, Powers) = \mathsf{true}, t)$$

In a similar manner we generate the permissions, obligations and sanctions of each participant. Tables IV and V present a subset of the information associated with each participant of the CNP. We have restricted attention to the powers, obliga-

Table V. Execution of the Specification of the CNP (Explorers).

Time	**agent** e_1	**agent** e_2
0		
1		
2	$pow(e_1, bid(e_1, c_1, RContent, 1))$	$pow(e_2, bid(e_2, c_1, RContent, 1))$
3	$pow(e_1, bid(e_1, c_1, RContent, 1))$	
4		
5		
6	$pow(e_1, bid(e_1, c_1, RNContent, 2))$	$pow(e_2, bid(e_2, c_1, RNContent, 2))$
7	$pow(e_1, bid(e_1, c_1, RNContent, 2))$	
8		
9		
10		$pow(e_2, inform(e_2, c_1, Result, 2))$, $obl(e_2, inform(e_2, c_1, Result, 2))$
11		$sanctioned(e_2)$

tions and sanctions of the agents. (Recall that, due to constraints (23) and (24), permissions in this CNP specification are the same as powers). In the interests of clarity and brevity, we have: (i) omitted the presentation of information related to explorer e_3 (in this run of the CNP, the powers, obligations and sanctions associated with e_3 are exactly the same as the powers, obligations and sanctions associated with e_1), and (ii) allowed the concurrent performance of CFPs.

At time-points $T = 0$ and $T = 1$, the three explorers do not have any powers associated with them. At time-point $T = 2$, each explorer is empowered to bid in regard to the cartographer's previous CFP (see constraint (20)). e_2 issues a valid bid at $T = 2$ (tasks $map(w)$ and $map(x)$ are 'related') and thus terminates its power to issue another bid (see constraint (21)). Due to the occurrence of $cTimeout$ at $T = 3$, neither e_1 nor e_3 is empowered to bid at $T = 4$ (see constraint (22)).

At $T = 4$, e_1 issues a bid; however, this bid is not valid because e_1 did not have the power to bid at the time. As a result, this invalid bid has no effect on the powers of the other participants of the CNP. At $T = 8$, the cartographer is not empowered to issue new CFPs because we have specified that the maximum protocol round is two.

At $T = 10$, explorer e_2 is obliged (in addition to being empowered) to report the result of $map(z)$, the awarded task. The obligation was created due to constraint (25). If e_2 had reported the result of $map(z)$ in the specified interval, then it would have discharged its obligation. e_2's failure to report the outcome of $map(z)$ by time $T = 10$ resulted in termination of its obligation (see constraint (26)) and imposition of a sanction at $T = 11$ (constraint (27)).

11. DISCUSSION

The CNP is not ordinarily presented in the literature in terms of the concepts of institutional power, permission, physical capability and sanction. Generally, work on the specification of open agent societies does not explicitly represent the

institutional powers of the member agents. This is one key difference between our work and related approaches in the literature: our specification of social constraints explicitly represents the institutional powers of the agents, differentiates between institutional power, permission and physical capability, and employs formalisms with a declarative semantics and clear routes to implementation to express these concepts.

The theoretical framework for the specification of open agent societies is not dependent on any particular action language or temporal structure. We have presented a specification of the CNP example both in terms of the $\mathcal{C}+$ language and in EC. Both have their relative merits.

The semantics of a $\mathcal{C}+$ action description is given in terms of a labelled transition system, and this provides a link to a wide range of other formalisms based on transition systems. This link may be exploited by combining $\mathcal{C}+$ with other software tools for the analysis of transition systems. In particular, we are currently experimenting with the use of standard model checking techniques [Clarke et al. 2000] to prove general properties of a society specification expressed in $\mathcal{C}+$. This point is developed further in [Sergot 2004a]. See also [Sergot 2004b] for an example.

It can also be argued that transition systems are not ideal for specifying open agent societies, for at least two reasons. First, transition systems have the property that actions executable in a state s, and their effects, depend only on s and not on the path by which s was reached. Thus, in our CNP specification it was necessary to record a part of the history in the states of the transition system, for instance including a record of all past valid bids in every state in order to determine whether a cartographer is empowered to award or reject bids. Second, transition systems provide no direct support for metrics of time. For example, there is no direct support for expressing statements of the form 'agent *ag* is permitted to perform action *act* (at the latest) by time t'. We implicitly specify this interval (both in our $\mathcal{C}+$ and EC specifications of the CNP) by stating the constraints that initiate and terminate the permission. Overcoming these limitations is an area for current research. (Craven and Sergot [2005] present a generalised form of $\mathcal{C}+$ designed to address, to different extents, these issues.)

The execution, as opposed to the specification, of social constraints may be performed either at design-time or at run-time. At design-time, society designers may wish to prove properties of a society specification in order to determine whether or not this specification meets their requirements. Such properties may include, for example, the consistency of a protocol specification (no agent is forbidden and obliged to perform an action at the same time), that in all circumstances there is always an obligation to report the outcome of an awarded task, that non-compliance with this obligation always leads to a sanction and possibly other reparational actions, and so on. Proving such properties can be performed, to some extent, via CCALC's computation of planning queries, as illustrated by the computations of Queries 2 and 4 in earlier sections. It should be noted, however, that the examples presented earlier contain a comparatively small number of agents; we have not determined in any systematic fashion to what extent CCALC can cope with planning queries on examples with very large numbers of agents.

In addition to the needs of society designers, agent designers may also wish to

prove various properties of a society specification when deciding whether to deploy their agents in that society. In some cases, it is not feasible for agent designers to examine in advance if they should deploy their agents in a society; the agents themselves may then be required to prove properties of a society specification before deciding whether or not to enter that society.

At run-time, the execution of a society specification may provide, amongst other things, information about the social state current at each time. Computation of such information is a special case of a prediction query. A social state — the powers, permissions, obligations and sanctions that are associated with each agent at each time — may be publicised to (a subset of) the members of a society, or their designers. (Such run-time services may be provided by a central server or in various distributed configurations. Further discussion of these architectural issues is outside the scope of this paper.) Both CCALC and SV compute prediction queries. Their efficiency in computing (such) queries, however, determines whether they are feasible for the provision of run-time services. We discuss CCALC's and SV's efficiency next.

CCALC's computation of a query has two phases:

—A *compilation phase* where the $\mathcal{C}+$ action description is translated to formulas of classical propositional logic.

—A *model generation phase* where a satisfiability (SAT) solver attempts to find models of the propositional logic formulas produced by the compilation phase that also satisfy the query.

In experiments on the action description D^{CNP}, and other similar examples, the time required for the computation of queries suggests that CCALC does not provide a practical means for supporting run-time activities. This is mainly due to the time required for compiling an action description. Details of execution times can be found in [Artikis et al. 2003b] and [Artikis 2003, Section 6.12.2]. Although parts of an action description can be compiled in advance, other parts change as, for instance, agents enter and leave a society, or (specifically in the CNP) new task descriptions are introduced, and this necessitates repeated re-compilation. Moreover, the construction of a model of an action description will necessarily generate a complete description of every social state at every point in time, which is obviously wasteful if we are interested only in determining (say) specific powers and permissions in the current (latest) state. It is possible to identify several ways of improving CCALC's efficiency for this type of application, for instance by identifying components of an action description that can be pre-compiled, or by attempting to limit the model that is generated to a fragment relevant to the query. Further development of these techniques is required. (Of course CCALC is not the only means by which $\mathcal{C}+$ action descriptions can be executed. Giunchiglia et al. [2004, Section 7.2] and Lifschitz and Turner [1999] show how $\mathcal{C}+$ action descriptions can be translated into the formalism of (extended) logic programs.)

For the CNP specification in Event Calculus, the computation of prediction queries using SV's implementation (in Prolog) was sufficiently fast for the anticipated run-time activities. Moreover, SV scaled better than CCALC, for example, when increasing the number of participants or protocol rounds in the example CNP specification. An indication of SV's times of query computation can be found in

[Artikis 2003, Section 6.12.2]. The existing Prolog implementation of EC in the SV system, moreover, does not include any optimisation techniques that can be devised for EC computations of prediction queries. Several such are available (see, for instance, [Chittaro and Montanari 1996; Kesim and Sergot 1996]). Incorporating such techniques would further improve SV's efficiency of query computation.

The implementation of EC action descriptions in Prolog, or indeed any other programming language[4], enables the specification of an action domain (a society or a protocol in this case) to employ features of the programming language to complement the use of EC itself. For example, the CNP specification in SV includes, apart from the EC axioms, logic rules (expressed in Prolog) that define when the task described in a bid is 'related' (similar) to that described in the CFP. Such a combination may: (i) result in a richer (more expressive) society or protocol specification and (ii) improve the efficiency of execution. It is much more natural and efficient, for example, to specify whether two tasks are 'related' by means of logic rules rather than EC axioms. The implementation of $\mathcal{C}+$ action descriptions in CCALC does not provide such facilities, and indeed it is difficult to see how it could do so, even in principle. Every aspect of a society or protocol specification, therefore, must be represented in terms of $\mathcal{C}+$ laws.

Future Work

There are several directions for current and future research. These include the following:

Alternative action languages. We presented a specification of the CNP by means of both $\mathcal{C}+$ laws and EC axioms. Intuitively, there is a close similarity between these formalisms, but establishing a precise equivalence remains an area for current work. One step in this direction is work by Craven and Sergot [2003] who have developed a method for translating (a subset of) $\mathcal{C}+$ laws to an equivalent EC-like formalisation, and thereby obtaining an efficient way of computing prediction queries on $\mathcal{C}+$ action descriptions.

The language $(\mathcal{C}+)^{++}$. A parallel development to the work presented in this paper has produced an extended form of the $\mathcal{C}+$ language specifically designed for the representation of norms and institutions [Sergot 2004a]. The intention is to extend the transition system structures with features that represent a 'counts as' relation between transition types (see Section 3.3), and, thereby, a treatment of institutionalised power, and the distinction between permitted and non-permitted actions and histories. Reformulation of the examples in this paper in $(\mathcal{C}+)^{++}$ is straightforward, except that there are some outstanding issues about the most effective way of employing the additional resources provided by $(\mathcal{C}+)^{++}$. In particular, the relationship between obligations, non-compliant behaviour and sanctions in society specifications, and the notion of permission built in to the $(\mathcal{C}+)^{++}$ language needs to be explored.

Functionality of the computational framework. Both CCALC and SV should provide explanatory facilities. A natural explanation of why, for instance, a given action

[4]See, for example, [Farrell et al. 2005] for an XML formalisation of EC, accompanied with a Java-implemented EC reasoner.

was or was not valid in some given circumstance would be an invaluable aid to the agents and their designers. In the case of SV, there are standard logic programming techniques for implementing explanatory facilities based on execution traces. In the case of CCALC, where the mode of execution (model generation) is quite different, it remains to investigate how explanatory facilities could be grafted on to the operation of CCALC.

Formalisation of normative relations. It can be argued that our treatment of the concepts of permission and obligation is too simple, and needs to be refined in the light of existing studies of these concepts. Moreover, we aim to extend our specification by considering more complex normative relations [Sergot 2001]. In particular, concepts of *entitlement* seem to be deserving of special attention. (See [Sadighi and Sergot 2002] for a preliminary discussion.)

Social structure. Social constraints should be relativised to a society. This would allow for the possibility that different societies (or institutions within the same society) have incompatible constraints. A fruitful area for further research would be to formalise the relationships between the social constraints of a 'parent society' and its sub-groupings (institutions).

ACKNOWLEDGMENTS

This paper is an updated and extended version of [Artikis et al. 2002; Artikis et al. 2003b]. We would like to thank the participants of the AAMAS '02 conference and the AOSE '02 workshop who gave us useful feedback. We are grateful to Rob Craven and Lloyd Kamara for their valuable comments on several drafts of this paper. We should also like to thank Joohyung Lee and Vladimir Lifschitz for their suggestions regarding the $\mathcal{C}+$ language and CCALC. The authors themselves, however, are solely responsible for any misunderstanding about the use of these technologies.

REFERENCES

AKMAN, V., ERDOGAN, S., LEE, J., LIFSCHITZ, V., AND TURNER, H. 2004. Representing the zoo world and the traffic world in the language of the Causal Calculator. *Artificial Intelligence 153,* 1–2, 105–140.

ARTIKIS, A. 2003. Executable specification of open norm-governed computational systems. Ph.D. thesis, University of London. Retrieved March 6, 2006, from `http://www.doc.ic.ac.uk/~aartikis/publications/artikis-phd.pdf`, also available from the author.

ARTIKIS, A., PITT, J., AND SERGOT, M. 2002. Animated specifications of computational societies. In *Proceedings of Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, C. Castelfranchi and L. Johnson, Eds. ACM Press, 1053–1062.

ARTIKIS, A., SERGOT, M., AND PITT, J. 2003a. An executable specification of an argumentation protocol. In *Proceedings of Conference on Artificial Intelligence and Law (ICAIL)*. ACM Press, 1–11.

ARTIKIS, A., SERGOT, M., AND PITT, J. 2003b. Specifying electronic societies with the Causal Calculator. In *Proceedings of Workshop on Agent-Oriented Software Engineering III (AOSE)*, F. Giunchiglia, J. Odell, and G. Weiss, Eds. LNCS 2585. Springer, 1–15.

BING, J. 1998. Managing copyright in a digital environment. In *The Impact of Electronic Publishing on the Academic Community*, I. Butterworth, Ed. Portland Press, 52–62.

BREWKA, G. 2001. Dynamic argument systems: a formal model of argumentation processes based on situation calculus. *Journal of Logic and Computation 11,* 2, 257–282.

Carley, K. and Gasser, L. 1999. Computational organization theory. In *Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence*, G. Weiss, Ed. MIT Press, 299–330.

Cevenini, C. 2003. Legal considerations on the use of software agents in virtual enterprises. In *The Law of Electronic Agents*, J. Bing and G. Sartor, Eds. Vol. CompLex 4/03. Oslo: Unipubskriftserier, 133–146.

Chittaro, L. and Montanari, A. 1996. Efficient temporal reasoning in the cached event calculus. *Computational Intelligence 12,* 3, 359–382.

Clark, K. 1978. Negation as failure. In *Logic and Databases*, H. Gallaire and J. Minker, Eds. Plenum Press, 293–322.

Clarke, E. M., Grumberg, O., and Peled, D. A. 2000. *Model Checking*. MIT Press, Cambridge.

Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F., and Jeremaes, P. 1994. *Object-Oriented Development: The FUSION Method*. Prentice Hall International.

Colombetti, M. 2000. A commitment-based approach to agent speech acts and conversations. In *Proceedings of Workshop on Agent Languages and Communication Policies*. 21–29.

Craven, R. and Sergot, M. 2003. Efficient computation for action languages. Tech. rep., Department of Computing, Imperial College, London.

Craven, R. and Sergot, M. 2005. Distant causation in C+. *Studia Logica 79,* 1, 73–96.

Davis, R. and Smith, R. 1983. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence 20*, 63–109.

Esteva, M., de la Cruz, D., and Sierra, C. 2002. ISLANDER: an electronic institutions editor. In *Proceedings of Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, C. Castelfranchi and L. Johnson, Eds. ACM Press, 1045–1052.

Esteva, M., Padget, J., and Sierra, C. 2002. Formalizing a language for institutions and norms. In *Intelligent Agents VIII: Agent Theories, Architectures, and Languages*, J.-J. Meyer and M. Tambe, Eds. LNAI 2333. Springer, 348–366.

Esteva, M., Rodriguez-Aguilar, J., Arcos, J., Sierra, C., and Garcia, P. 2000. Institutionalising open multi-agent systems. In *Proceedings of the International Conference on Multi-agent Systems (ICMAS)*, E. Durfee, Ed. IEEE Press, 381–382.

Esteva, M., Rodriguez-Aguilar, J., Sierra, C., Garcia, P., and Arcos, J. 2001. On the formal specifications of electronic institutions. In *Agent Mediated Electronic Commerce*, F. Dignum and C. Sierra, Eds. LNAI 1991. Springer, 126–147.

Farrell, A., Sergot, M., Sallé, M., and Bartolini, C. 2005. Using the event calculus for tracking the normative state of contracts. *International Journal of Cooperative Information Systems 4,* 2–3, 99–129.

Ferber, J. and Gutknecht, O. 1998. A meta-model for the analysis and design of organizations in multi-agent systems. In *Proceedings of International Conference on Multi-Agent Systems (ICMAS)*, Y. Demazeau, Ed. IEEE Press, 128–135.

Ferber, J. and Gutknecht, O. 2000. Operational semantics of multi-agent organisations. In *Intelligent Agents IV: Agent Theories, Architectures, and Languages*, N. Jennings and Y. Lesperance, Eds. LNCS 1757. Springer, 205–217.

Fitoussi, D. and Tennenholtz, M. 2000. Choosing social laws for multi-agent systems: minimality and simplicity. *Artificial Intelligence 119,* 1-2, 61–101.

Foster, I., Kesselman, C., and Tuecke, S. 2001. The anatomy of the grid — enabling scalable virtual organisations. *International Journal of Supercomputer Applications 15,* 3, 200–222.

Fox, M., Barbuceanu, M., Grüninger, M., and Lin, J. 1998. An organizational ontology for enterprise modeling. In *Simulating Organizations: Computational Models for Institutions and Groups*, M. Prietula, K. Carley, and L. Gasser, Eds. AAAI Press/The MIT Press, 131–152.

Gelati, J., Governatori, G., Rotolo, A., and Sartor, G. 2002. Declarative power, representation, and mandate. A formal analysis. In *Proceedings of Conference on Legal Knowledge and Information Systems (JURIX)*.

Gelati, J., Rotolo, A., and Sartor, G. 2002. Normative autonomy and normative coordination: declarative power, representation, and mandate. In *Proceedings of Workshop on the Law of Electronic Agents (LEA)*. 133–149.

Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., and Turner, H. 2004. Nonmonotonic causal theories. *Artificial Intelligence 153,* 1–2, 49–104.

Giunchiglia, E., Lee, J., Lifschitz, V., and Turner, H. 2001. Causal laws and multi-valued fluents. In *Proceedings of Workshop on Nonmonotonic Reasoning, Action and Change (NRAC)*.

Governatori, G., Gelati, J., Rotolo, A., and Sartor, G. 2002. Actions, institutions, powers: preliminary notes. In *Proceedings of Workshop on Regulated Agent-Based Social Systems: Theories and Applications (RASTA)*. 69–79.

Hardwick, M. and Bolton, R. 1997. The industrial virtual enterprise. *Communications of the ACM 40,* 9, 59–60.

Hewitt, C. 1991. Open information systems semantics for distributed artificial intelligence. *Artificial Intelligence 47*, 79–106.

Jones, A. 2001. Roles. In *Deliverable D2 of ALFEBIITE EU-Project (IST-1999-10298)*, A. Jones and C. Krogh, Eds. 52–55. Retrieved March 6, 2006, from `http://alfebiite.ee.ic.ac.uk/docs/Deliverables/D2.pdf`.

Jones, A. and Sergot, M. 1993. On the characterisation of law and computer systems: the normative systems perspective. In *Deontic Logic in Computer Science: Normative System Specification*. J. Wiley and Sons, 275–307.

Jones, A. and Sergot, M. 1996. A formal characterisation of institutionalised power. *Journal of the IGPL 4,* 3, 429–445.

Kautz, H. and Selman, B. 1992. Planning as satisfiability. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*, B. Neumann, Ed. J. Wiley and Sons, 359–363.

Kesim, F. and Sergot, M. 1996. Implementing an object-oriented deductive database using temporal reasoning. *Journal of Database Management 7,* 4.

Kowalski, R. and Sergot, M. 1986. A logic-based calculus of events. *New Generation Computing 4,* 1, 67–96.

Lee, J., Lifschitz, V., and Turner, H. 2001. A representation of the zoo world in the language of the Causal Calculator. In *Proceedings of Symposium on Formalizations of Commonsense Knowledge*.

Lifschitz, V. 2000. Missionaries and cannibals in the Causal Calculator. In *Proceedings of Conference on Principles of Knowledge Representation and Reasoning (KR)*, A. Cohn, F. Giunchiglia, and B. Selman, Eds. Morgan Kaufmann, 85–96.

Lifschitz, V., Mccain, N., Remolina, E., and Tacchella, A. 2000. Getting to the airport: the oldest planning problem in AI. In *Logic-Based Artificial Intelligence*, J. Minker, Ed. Kluwer, 147–168.

Lifschitz, V. and Turner, H. 1999. Representing transition systems by logic programs. In *Proceedings of Fifth International Conference on Logic Programming and Nonmonotonic Reasoning*. 92–106.

Makinson, D. 1986. On the formal representation of rights relations. *Journal of Philosophical Logic 15*, 403–425.

Manna, Z. and Pnueli, A. 1992. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag.

Manna, Z. and Pnueli, A. 1995. *Temporal Verification of Reactive Systems — Safety*. Springer-Verlag.

Minsky, N. and Ungureanu, V. 2000. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *ACM Transactions on Software Engineering and Methodology (TOSEM) 9,* 3, 273–305.

Moses, Y. and Tennenholtz, M. 1992. On computational aspects of artificial social systems. In *Proceedings of Workshop on Distributed Artificial Intelligence (DAI)*. 267–284.

Moses, Y. and Tennenholtz, M. 1995. Artificial social systems. *Computers and Artificial Intelligence 14,* 6, 533–562.

Noda, I., Matsubara, H., Hiraki, K., and Frank, I. 1998. Soccer server: a tool for research on multi-agent systems. *Applied Artificial Intelligence 12,* 2–3, 233–250.

Pinto, J. and Reiter, R. 1993. Temporal reasoning in logic programming: a case for the situation calculus. In *Proceedings of Conference on Logic Programming*, D. Warren, Ed. MIT Press, 203–221.

Pitt, J., Kamara, L., and Artikis, A. 2001. Interaction patterns and observable commitments in a multi-agent trading scenario. In *Proceedings of Conference on Autonomous Agents (AA)*, J. Müller, E. Andre, S. Sen, and C. Frasson, Eds. ACM Press, 481–489.

Pörn, I. 1977. Action theory and social science: some formal models. In *Number 120 in Synthese Library*. D. Reidel Publishing Company.

Prakken, H. 2005. Coherence and flexibility in dialogue games for argumentation. *Journal of Logic and Computation 15*, 1009–1040.

Prakken, H. and Gordon, T. 1999. Rules of order for electronic group decision making – a formalization methodology. In *Collaboration between Human and Artificial Societies*, J. Padget, Ed. LNCS 1624. Springer, 246–263.

Rodriguez-Aguilar, J. and Sierra, C. 2002. Enabling open agent institutions. In *Socially Intelligent Agents: Creating relationships with computers and robots*, K. Dautenhahn, A. Bond, L. Canamero, and B. Edmonds, Eds. Kluwer Academic Publishers, 259–266.

Rosenschein, J. and Zlotkin, G. 1994. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. The MIT Press.

Sadighi, B. and Sergot, M. 2002. Contractual access control. In *Proceedings of Workshop on Security Protocols*. LNCS 2845. Springer, 96–102.

Santos, F., Jones, A., and Carmo, J. 1997. Action concepts for describing organised interaction. In *Hawaii International Conference on System Sciences (HICSS)*, R. Sprague, Ed. C.S. Press, 373–382.

Searle, J. 1969. *Speech Acts*. Cambridge University Press.

Sergot, M. 2001. A computational theory of normative positions. *ACM Transactions on Computational Logic 2,* 4, 522–581.

Sergot, M. 2004a. $(\mathcal{C}+)^{++}$: An action language for modelling norms and institutions. Tech. Rep. 2004/8, Department of Computing, Imperial College London. Retrieved March 6, 2006, from `http://www.doc.ic.ac.uk/research/technicalreports/2004/DTR04-8.pdf`.

Sergot, M. 2004b. Modelling unreliable and untrustworthy agent behaviour. In *Proceedings of Workshop on Monitoring, Security, and Rescue Techniques in Multiagent Systems (MSRAS)*, B. Dunin-Keplicz, A. Jankowski, A. Skowron, and M. Szczuka, Eds. Advances in Soft Computing. Springer-Verlag, 161–178.

Shoham, Y. and Tennenholtz, M. 1992. On the synthesis of useful social laws for artificial agent societies. In *Proceedings of Conference on Artificial Intelligence*. The AAAI Press/ The MIT Press, 276–281.

Shoham, Y. and Tennenholtz, M. 1995. On social laws for artificial agent societies: off-line design. *Artificial Intelligence 73,* 1-2, 231–252.

Singh, M. 1998. Agent communication languages: rethinking the principles. *IEEE Computer 31,* 12, 40–47.

Singh, M. 2000. A social semantics for agent communication languages. In *Issues in Agent Communication*, F. Dignum and M. Greaves, Eds. LNCS 1916. Springer, 31–45.

Sirbu, M. 1997. Credits and debits on the Internet. *IEEE Spectrum 34,* 2, 23–29.

Smith, R. 1980. The contract-net protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers 29,* 12, 1104–1113.

Smith, R. and Davis, R. 1978. Distributed problem solving: the contract-net approach. In *Proceedings of Conference of Canadian Society for Computational Studies of Intelligence*. 217–236.

Tennenholtz, M. 1995. On computational social laws for dynamic non-homogeneous social structures. *Journal of Experimental and Theoretical Artificial Intelligence 7*.

Werner, E. 1989. Cooperating agents: a unified theory of communication and social structure. In *Distributed Artificial Intelligence*, L. Gasser and M. Huhns, Eds. Vol. II. Morgan Kaufmann, 3–36.

Werner, E. 1992. The design of multi-agent systems. In *Decentralized A.I.*, E. Werner and Y. Demazeau, Eds. Vol. 3. Elsevier Science Publishers, 3–30.

Wooldridge, M., Jennings, N., and Kinny, D. 1999. A methodology for agent-oriented analysis and design. In *Proceedings of Conference on Autonomous Agents (AA)*, O. Etzioni, J. Müller, and J. Bradshaw, Eds. ACM Press, 69–77.

Wooldridge, M., Jennings, N., and Kinny, D. 2000. The Gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems 3,* 3, 285–312.

Zambonelli, F., Jennings, N., Omicini, A., and Wooldridge, M. 2001. Agent-oriented software engineering for internet applications. In *Coordination of Internet Agents: Models, Technologies, and Applications*, A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, Eds. Springer, 326–346.

Zambonelli, F., Jennings, N., and Wooldridge, M. 2001a. 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.

Zambonelli, F., Jennings, N., and Wooldridge, M. 2001b. Organizational abstractions for the analysis and design of multi-agent systems. In *Agent-Oriented Software Engineering*, P. Ciancarini and M. Wooldridge, Eds. LNCS 1957. Springer, 235–252.

Zambonelli, F., Jennings, N., and Wooldridge, M. 2003. Developing multiagent systems: The gaia methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM) 12,* 3, 317–370.