Guaranteeing Properties for E-Commerce Systems

Frank Guerin and Jeremy Pitt

Intelligent and Interactive Systems, Department of Electrical & Electronic Engineering,
Imperial College of Science, Technology & Medicine, Exhibition Road, London, SW7 2BT.
{f.guerin,j.pitt}@ic.ac.uk, Phone: +44 20 7594 6318 / Fax: 6274

Abstract. Agents in an electronic commerce system act on behalf of (potentially) competing individuals and organisations. It is proposed that such systems will be used in scenarios where legally binding contracts are made or money is exchanged by the agents on behalf of their owners. Agent owners may be reluctant to delegate tasks involving uncertain and possibly detrimental outcomes to an agent without assurances about the system's properties. It may be a requirement, for example, that an agent cannot profit from lying to its peers. This paper demonstrates how solutions from game theory together with computing theories can be used to publicly specify rules and prove desirable properties for agent systems. This has the potential to increase the range of applications in which agent owners may be willing to delegate to their embedded counterparts.

1 Motivation

Agents in an electronic commerce system may be developed and owned by different individuals or organisations who may have conflicting interests; hence the internals (i.e. program and state) of agents are not public and so notions of trust and deception are relevant; we call this an open system. It is proposed that such systems will be used in scenarios where legally binding contracts are made and money is exchanged by the agents on behalf of their owners. Not surprisingly, agent owners can be expected to be reluctant to delegate tasks involving potentially detrimental outcomes to an agent unless they can be assured that the system has certain desirable properties. For example, does the system guarantee that my agent will not be discriminated against in favour of my competitor? Or that my competitor cannot benefit by lying to my agent? Or that I get the optimal price? It may be impossible to guarantee the most desirable outcomes for all participants, but the system should be at least as good as the best non-agent alternative. The best alternatives are real market mechanisms. Self interested rational agents in an open society can be treated in much the same way as humans playing games or participating in markets. Solutions from game theory and economics [3] allow us to design mechanisms for interactions which have the properties we desire. To prove that these properties hold for agent systems we need to examine the details of their implementation with computing theories. This paper aims to bridge the gap between the solutions which game theory provides for systems of self interested agents and the underlying computational processes which implement the agents in an electronic commerce system. This has the potential to increase the range of applications in which agent owners may be willing to delegate to their embedded counterparts.

2 Related Work: Engineering Interactions

There are many scenarios in which self-interested rational individuals interact to come to some agreement; for example selling goods, awarding contracts or negotiating deals. Such interactions are typically constrained by some rules (the mechanism) and within these constraints agents will plan their best course of action (the strategy). A mechanism is a set of public rules governing an interaction, for example an auction mechanism can be used to sell an item. These public rules physically constrain the actions of the participants and moreover, they can be designed to induce certain outcomes; for example an auction can be designed so that the seller obtains the highest possible price or so that the bidders bid truthfully. In these cases the mechanism is not forcing participants to produce the desired outcome, it is ensuring that it is in the participants' interest to take the actions it is trying to induce. Mechanisms do this by making the rules by which the agreement is reached public. Thus the participants can unambiguously determine the consequences of their actions in advance and determine their best course of action (which will naturally be one of the actions the mechanism designer intended to induce).

There are a number of desirable properties for mechanisms [12, 11]:

- *Efficiency*. The most efficient mechanism will maximise the social welfare which is the sum of all the agents' utilities. Pareto efficient mechanisms have the property that no agent could do better from another mechanism without some other agent doing worse; this is probably a more useful criterion because it does not require inter-agent utility comparisons [12].
- *Individual Rationality*. An agent who participates in the mechanism should derive a utility at least as large as that obtained by not participating [3]. Otherwise a self interested agent would not participate.
- *Incentive Compatibility*. The mechanism must provide incentives that make it optimal for the agents to take the actions that the mechanism designer is intending to induce [3]. For example, if we want agents to tell the truth we must design the mechanism so that agents do not gain any advantage by lying, such a mechanism is called *deception free*. In general a mechanism cannot ensure that agents do not deceive, but it can be designed so that it is in their interest not to.
- *Stability*. Agents should not have to change strategies because of other agents' actions. Ideally there will be a dominant strategy that is best for each agent regardless of what other agents do. Nash equilibrium is achieved when an agent uses a certain strategy and competing agents cannot do better by using a different strategy. In this way mechanisms can eliminate the need for agents to engage in complicated reasoning to determine their best course of action [12]. A strategy which is not just beneficial to society, but to each individual too, is resistant to outside invasion [1].
- *Symmetry*. The mechanism should not favour or be biased against any particular agent. Agents that act alike should receive the same utility [12].
- *Simplicity*. A simple mechanism will have little communication overhead and few resources spent outguessing, leading to low computational demands.
- *Distribution*. It may be desirable not to rely on a centralised decision maker.

Most of the existing work on applying game theory and economics to problems in agent systems can be grouped in the following three categories:

1. Mechanism design. For example Parkes [9] looks at auction design with particular attention to an agent's computational limitations; Sandholm [12] shows how interactions can be broken down into chunks so that enforcement is not necessary to encourage participants to complete a transaction.
2. Strategy design and algorithms. For example bidding strategies in an auction [2] or algorithms for winner determination [13].
3. Low level computing theories required to formally specify and implement a mechanism for a system of agents, and to verify that the system complies with the specification. For example van Eijk [5] shows how an agent can be given a formal specification to implement the Zeuthen strategy; Pradella and Colombetti [10] show how correctness can be proved for BDI agents engaged in trading.

We are concerned with third group: the low level computing theories required to formally specify a mechanism and to guarantee that a system of agents using it does indeed have the desired properties. We use computing theories to guarantee behaviours orthogonal to those that game theory seeks to guarantee; the computing theories cannot guarantee that an agent will be truthful, but if the game theory properties ensure that the agent gains the highest expected utility by being truthful and if the computing theories guarantee that the game theory mechanism is adhered to, then a rational agent will be truthful. Ensuring that the mechanism is adhered to means, for example, that the auctioneer in a second price auction does sell the lot to the highest bidder (and not some other agent) and that the price charged is the second highest bid (and not a fictitious bid inserted by the auctioneer). To this end, we must provide a method by which the formal specification of a mechanism can be made public; we believe this is necessary so that:

1. All participants can inspect it and verify its properties for themselves at design time.
2. Participants' actions can be analysed to ensure that they do follow it at run time.

It is important that the participants can be assured of the mechanism being used so that they know that their optimal strategy is to bid truthfully, for example, then the agent designer need not consider deceptive strategies. If a mechanism is designed to be incentive compatible then it goes without saying that the participants must be aware that this is the case; for example, it is not sufficient that a winner determination algorithm be coded solely in the internals of an auctioneer agent.

The frameworks of Van Eijk's [5] and Pradella and Colombetti [10] do not fit our requirements because they requires knowledge of an agent's internal information store which may not be available in an open system. Work on e-institutions [6] does allow a formal specification of the public rules for a protocol; however this work stops short of giving the rules of the institution with reference to some computational model. In particular, normative rules are specified with an *obliged* predicate which has not been given a semantics relative to a grounded model; this leaves its interpretation unclear. A computational model is necessary to provide a reference which allows agents' actions to be compared with the rules of the institution and thus enables verification. To the best of our knowledge the only framework which would allow the specification and verification of protocols in open systems is Singh's [15]. The main difference between this and our approach is that we will use a denotational semantics which gives a procedural interpretation to communications while Singh's is declarative.

3 Formal Framework

Verification typically involves comparing the agents' behaviours with the behaviour mandated by the rules of the system. We need to formalise the states and programs of agents if we wish to know their exact behaviour [16]. We also formalise the externally observable actions of agents to allow us to verify their behaviour in the absence of internal information (in an open system). We represent states and programs of agents with a computational model. The only action we consider in this paper is communication, thus the rules of the system are specified entirely through an agent communication language (ACL) which defines the semantics for communicative actions.

3.1 Agent Programs

We treat a multi-agent system as a reactive program [8]. The entire multi-agent system can be described by a single program $P :: \|_{i \in Ag} M_i$. This is a cooperation statement (for parallel execution) where each top level process M_i is viewed as a module which represents the program of agent i, where i ranges across the set of agent names Ag which uniquely identify agents. Each module M_i will declare the data variables it uses and will contain the statements that describe its behaviour (in some programming language). Among the variable declarations of each module are sets of asynchronous buffered input and output channels on which messages can be exchanged with other agents. A channel is a variable whose value is a list of messages. We identify channel variables using an α with a subscript; agent i receives messages from agent j on the channel $\alpha_{i,j}$ and sends them on channel $\alpha_{j,i}$. The asynchronous communication predicate $[\alpha \Leftarrow m]$ denotes a sending event i.e. when a message is sent to a channel α; it is defined as $\neg \mathit{first} \ \wedge \ \alpha = \alpha^{-} \bullet m$. That is, a sending event has occurred on channel α if this is not the first state and if variable α is equal to what it was before (α^{-}) with message m appended to the end of the list.

3.2 Computational Model

A computational model is a mathematical structure which can represent the behaviour of a program. In this section we review Manna and Pnueli's computational model for reactive programs [8] and apply it to multi agent systems. In the next section we make some extensions to capture observable states of an open system. The computational model is a fair transition system; it contains variables and transitions. Variables represent the states of the agents and transitions represent the state changes caused by statements in the agents' programs. The variables come from a universal set of typed variables $\mathcal{V}$, called the vocabulary; from this we can construct assertions (e.g. $\forall x : \exists y : \ y > x$). A *state* s is an interpretation of $\mathcal{V}$, assigning each variable $u \in \mathcal{V}$ a value $s[u]$ over its domain. The transitions in the system tell us how one state can move to the next. A transition is *taken* at state s if the next state is related to s by the transition. Our multi-agent system is represented by the fair transition system $S = \langle V, \Theta, \mathcal{T}, \mathcal{J}, \mathcal{C} \rangle$ which represents the states and programs of the agents in the system.

- $V \subseteq \mathcal{V}$ is a set of system variables, these represent data variables (i.e. data within an agent's internal state) and the location of control within an agent's program.

– Θ is an assertion characterising initial states.
– $\mathcal{T}$ is a set of transitions. A transition maps each state onto a set of possible successor states. Each statement in an agent's program is associated with a transition in $\mathcal{T}$.
– $\mathcal{J}, \mathcal{C} \subseteq \mathcal{T}$ are the sets of just and compassionate transitions, they ensure that each parallel process is executed fairly (more detail in [8]).

The agent programs which constitute our multi-agent system must be given a semantics which identifies each of the components of the fair transition system; this can be done using the Simple Programming Language (SPL) [8]. An infinite sequence of states $\sigma : s^0, s^1, s^2, s^3, \ldots$ is called a *system model*. A system model is a *computation of the program* P (which identifies our fair transition system) if s^0 satisfies the initial condition Θ and if each state s^{j+1} is accessible from the previous state s^j via one of the transitions $\mathcal{T}$ in the system and the requirements of justice and compassion are respected. A computation is a sequence of states that could be produced by an execution of the program. All computations are system models but not vice versa.

3.3 Agent Communication Language and Observable States

As described in previous work [7] we advocate the use of a *social* ACL for open systems. This is because agents' internal states may not be accessible so the most suitable approach for verification lies with an ACL which is based on publicly observable phenomena. A message m which is a well formed formula of our communication language $\mathcal{L}_c$ is a 4-tuple including the sender, receiver, speech act name and content. The message m is given a semantics $[\![m]\!]_c$ which defines a change to the observable *social state*.

$$[\![-]\!]_c \ : \ \mathit{wff}(\mathcal{L}_c) \rightarrow (\Omega \rightarrow \Omega)$$

Where Ω is the set of all observable social states. The observable social states are described by a set of variables that is disjoint from the fair transition system variables V. The social state describes *social facts* using a language $\mathcal{L}_f$ and includes role relationships, control variables, commitments to perform actions and publicly expressed attitudes. Each agent i may have a differing view of the social context, since it may not have received all communications occurring in the system. We use the variable o_i to denote the social state observable to agent i. A social state o_i is a tuple $\langle p_i, c_i \rangle$ where p_i is the state of *persistent* social facts (dealing with long term commitments for example) and c_i is the *contingent* social facts for the current conversation. Contingent facts are solely concerned with the current conversation; for example a commitment to reply to a request in a conversation is a contingent commitment, but the commitment of the winner of an auction to buy an item is persistent. Persistent facts persist until revoked; contingent facts depend on the current state and will be re-evaluated after each act. The variables c_i and p_i map formulae of the social facts language to boolean values.

$$c_i, p_i : \ \mathit{wff}(\mathcal{L}_f) \rightarrow \{true, false\}$$

Each social fact x for an agent i is given a semantics $[\![x, i]\!]_f$ which describes (via temporal logic) the set of models where the agent i has satisfied x.

$$[\![-]\!]_f \ : \ (\mathit{wff}(\mathcal{L}_f) \times Ag) \rightarrow \mathit{wff}(\mathcal{L}_t)$$

Where $\mathcal{L}_t$ is the language of linear temporal logic [8]. The social facts semantics function $[\![-]\!]_f$ is allowed to make use only of channel variables and the observable state variables o_i so it cannot place constraints on agent internals. Many social facts may be simply satisfied in all situations, but some such as a commitment to do an action may be satisfied only in those models where the action is eventually done by the agent.

We define an initial condition Φ for the social state variables to specify role relationships and/or commitments existing when the system starts. We stated above the conditions which a system model σ must satisfy in order to be considered a *computation of the program* P; this constrained how the states in σ could interpret the variables in V but not variables outside V. We now define a *computation of the multi-agent system* S with initial social fact Φ to be a system model $\sigma : s^0, s^1, s^2, s^3, \ldots$ which is a computation of the program P and additionally s^0 satisfies Φ and whenever a transition is taken which involves message passing, the social facts variables are updated accordingly. This update means that for each state s^{j+1} which has been reached from the previous state s^j via a message passing transition, the social state o_i for both sender and receiver must be updated by the semantics $[\![-]\!]_c$ defined by the ACL; i.e. the new value o_i is equal to $[\![m]\!]_c o_i^-$ where m is the message passed and o_i^- is the old value at s^j. We have shown how the function $[\![-]\!]_c$ can be formally specified in previous work [7].

Note that if we can observe communication and are given an initial value for social facts, we can work out the subsequent values; they do not capture any additional information and are merely a convenience which we will use when proving that an agent satisfies the social facts which it can observe. Because of this redundancy the social state is not a part of our fair transition system; it need not be explicitly represented anywhere in a real multi-agent system, parts of it may or may not be represented by the local variables of agents in the system. Each agent should store a copy of all the information in the social state that might be relevant to its interactions so that it may correctly interpret context dependent communicative acts and keep track of its social commitments. As an external observer we need to know the social state if we wish to understand the interactions taking place in the system; we need to know the social knowledge observable to each agent in a system in order to determine if it is complying with the social conventions to the best of its knowledge.

3.4 External System

If we do not have access to the internals of any agent, but can detect each message being sent, we can construct a fair transition system S_E which represents all possible observable sequences. The variables of S_E are simply the communication channels we can observe and there are only two transitions, the idling transition τ_I (preserves all variables and does nothing) and the environmental transition τ_E. The environmental transition allows arbitrary modification of any variable outside of the observable channels and allows a channel in $\mathcal{A}$ to be modified by adding a message to the end or removing one from the front. The initial condition Θ requires that all channels are empty. In order to complete the social states in computations of the multi-agent system S_E we must also know the initial social facts for the initial condition Φ. A computation of the multi-agent system S_E does not care about how its states interpret variables which are not observable, it only cares about channels and social facts variables.

4 Verification For Open Systems

The possible types of verification depend on the *information available* and whether we wish to verify at *design time* or at *run time*. We look at three types.

Type 1. *Verify that an Agent Will Always Comply*

If we have access to an agent's internals (for example we are the agent designers) we can verify at design time that the agent will always respect its social commitments regardless of what other agents in a system do. From the agent's code we construct the transition system S_i which represents all the possible behaviours of agent i in any system (see [8] page 38). Then we prove that in all computations of the multi-agent system S_i, whenever a social fact x is mapped to true by f_i at a state, then the semantics $[\![x, i]\!]_f$ of that fact holds in the model. If agent i is implemented by a finite state program[1] we can use a model checking algorithm to perform the verification, it is less complex than proof theoretic verification.

Type 2. *Verify Compliance by Observation*

Here we determine if an agent i is compliant by observing its behaviour at run time; we assume that we do not have access to the internal state of the agent. We must have access to an initial description of social relations Φ and the observed communications involving i. We construct a finite sequence of social states σ_h which is consistent with the sequence of the observed communications. We construct a fair transition system S_E which represents all possible observable sequences of states. Φ is the initial condition of S_E, the channels are the variables and the transitions are τ_I and τ_E as described in section 3.4. We can now construct the set of possible models of our observed sequence: these models are computations of the multi-agent system S_E which match the observed finite sequence, thereafter they take all possible paths by taking the idling or environmental transitions. Our task then is to verify if there exists a possible model of our observed sequence in which each social fact x that is true for agent i, at any state, has its semantic formula $[\![x, i]\!]_f$ satisfied in the model.

Type 3. *Verify Protocol Properties*

Proving properties of protocols at design time is possible using only the ACL specification. Essentially we prove that the property holds for a system of compliant agents executing the protocol. We construct a fair transition system S_E which represents all possible observable sequences of states (see section 3.4) and we let the initial condition Φ be an assertion characterising a social state where the protocol has started. Then we prove that the desired property holds over all computations of the multi-agent system S_E. We demonstrate this type of verification in section 8.

[1] In a finite state program each variable assumes only finitely many values in all computations.

4.1 Policing The Society

Our verification of type 3 requires that all agents comply. To be able to use this in an e-commerce system there must be some way to enforce compliance. Verification type 2 is important because we must be able to identify misbehaving agents if we are to take action against them (sanctions) and hence guarantee that they will not prevent the society from functioning in the desired way. Policing a society can be tackled by:

1. Sentinel agents may monitor the observable behaviour of agents and have the capability to place sanctions or to evict or terminate offending agents.
2. The society can police itself: we may introduce notions like politeness, whereby agents violating certain commitments are branded as antisocial and are ostracised by the rest of the society. Prisoner's dilemma experiments [1] have shown that a strategy of reciprocating has the effect of policing the society because agents will not tend to misbehave if they do not thereby gain an advantage.
3. Agent owners will be legally responsible for the behaviour of their agents. Agents will not be allowed to participate in a system unless their owner guarantees that they are compliant (the agent designer can do this with verification type 1). If such an agent misbehaves at run time some action can be taken against the agent owner.

With enforcement we can for example enforce the requirement that the auctioneer should sell at the second highest price in a Vickrey auction if a sentinel agent monitors all transactions. This does not mean that agents' bids need to be revealed to each other in a sealed bid auction; the sentinel agent may be a trusted third party who can inspect all bids. This enforcement contrasts with the approach of Sandholm [12] who designs mechanisms for scenarios where no sort of enforcement is possible; also, Rosenschein and Zlotkin [11] advocate designing the mechanism so that social conventions are stable, because they are individually motivated. We believe that making enforcement possible increases the range of possible applications of e-commerce agents.

5 Example: A Simple Auction

We now look at a simple scenario where an auctioneer wishes to sell an item for the highest possible price [3]. There are two bidders who each privately value the item at either 3 or 4. We call a bidder with a high valuation a high agent, and one with a low valuation is a low agent. It is common knowledge that the probability of an agent being high is $\frac{1}{2}$. We assume participants are risk neutral, i.e. they are indifferent when faced with a choice of receiving a guaranteed sum of money x or participating in a game where the expected value of their winning is x. We also assume the bidders make private valuations, i.e. they know their own valuation for the item and it is independent of the other agent's valuation. The auctioneer wishes to get the largest price possible. Since the Auctioneer does not know the bidders' true valuations she cannot obtain the first best outcome (i.e. sell it to the highest agent for his true valuation). In searching for an optimal mechanism we need only consider direct mechanisms (where agents declare their true type, low or high) since the revelation principle tells us that whatever can be done with an indirect mechanism can also be done with a direct one. The auction

mechanism which is optimal for the seller in this auction has been analysed by Binmore [3], it is a modified Vickrey auction. A direct mechanism is employed where the bidders declare their type, if both bidders declare high or both declare low the winner is determined randomly with each having a probability of $\frac{1}{2}$ to win. In the case where one bidder bids high and the other bids low, the high bidder always wins. The price paid by a winning low bidder is 3 and a winning high bidder pays $3\frac{2}{3}$. Losing bidders pay nothing. Interesting properties to prove for an auction include

- Feasibility, i.e. the sum of the probabilities of each agent winning should not exceed unity. Also incentive compatibility and individual rationality should hold.
- Optimal expected value for the seller.
- Symmetry, i.e. the probability of an agent winning (probability that he wins if he bids high plus probability of winning if he bids low) should be equal for each agent.

An auction mechanism has the property of *incentive compatibility* if the bidders do not have an incentive to lie. It has *individual rationality* if the bidders are not better off by simply not participating. To see that these properties hold for the mechanism consider first the case of the low agent. There is no incentive for a low agent to bid high as doing so would mean a chance of obtaining negative utility by paying more than his valuation for the item and no chance of obtaining positive utility. There is still an incentive for the low agent to participate since he may win and pay his true valuation for the item. However the price a low agent pays when he wins must not be any greater than 3 or else it would not be individual rational to participate. A high agent always wins the auction if his opponent is a low agent (the probability of this is $\frac{1}{2}$) and wins half the time if his opponent is high. So his probability of winning against the unknown opponent is $\frac{3}{4}$. If he wins he pays $3\frac{2}{3}$ even though the item is worth 4 to him, so his utility is $\frac{1}{3}$. Therefore, by telling the truth he gains utility $\frac{1}{3}$ with probability $\frac{3}{4}$ and the expected value of his utility is $\frac{1}{3} \times \frac{3}{4} = \frac{1}{4}$. If he lies and bids low he reduces his probability of winning to $\frac{1}{4}$ but if he wins he pays only 3 for the item he values at 4, thereby gaining utility 1. Therefore by lying the expected value of his utility is $1 \times \frac{1}{4} = \frac{1}{4}$, i.e. the same. So $3\frac{2}{3}$ is as high as the auctioneer can make the price for a high agent without giving the high agent an incentive to bid low. This completes the description of the mechanism's properties. We now specify the protocol for an agent system and then show that it has these properties.

6 Specification of Protocol Properties

We must now specify the desired protocol properties with temporal formulae which describe the outcome that will result from whatever strategies the participants choose. In our case, to show a participating agent what he expects to pay if he wins when he bids high or low. We identify our agents as $Ag = \{ag1, ag2, ag3, rand\}$; these are bidder 1, bidder 2, auctioneer and *rand*, an agent who will decide the winner randomly when both agents declare the same value. The code of the *rand* agent should be open source so that the fairness of its algorithm can be verified (not attempted here). We must create an initial condition Φ which characterises a social state where the protocol has just begun. After this the next action of the bidders should be to bid high or low; if the bids are the same, the agent *rand* will then be called upon. Finally the auctioneer awards the lot and

announces that the auction is over. We create another assertion Ψ which characterises all terminal states of the protocol. Once Φ is true, Ψ will remain false until the termination of the protocol, at which time it becomes true.

Now we design the agents' strategies. For each agent we must include a strategy which encodes every possible sequence of choices that the agent can make during the protocol. We need two strategies for each of our bidding agents and for agent *rand* too, it also has two possible actions. Let *next*(i, m) stand for a temporal formula which is true at a state in a model if m is the next message sent by agent i and m is sent before termination of the protocol. Agents use the performative *tell* using a content 'H' or 'L' for the bidders and '1' or '2' for *rand*. Here are the possible strategies for all agents:

$$\begin{aligned}
\mathcal{S}_1 &: \ \Phi \rightarrow \textit{next}(\textit{ag1}, (\textit{ag1}, \textit{ag3}, \text{tell}, \text{H})) \\
\mathcal{S}_2 &: \ \Phi \rightarrow \textit{next}(\textit{ag1}, (\textit{ag1}, \textit{ag3}, \text{tell}, \text{L})) \\
\mathcal{S}_3 &: \ \Phi \rightarrow \textit{next}(\textit{ag2}, (\textit{ag2}, \textit{ag3}, \text{tell}, \text{H})) \\
\mathcal{S}_4 &: \ \Phi \rightarrow \textit{next}(\textit{ag2}, (\textit{ag2}, \textit{ag3}, \text{tell}, \text{L})) \\
\mathcal{S}_5 &: \ \Phi \rightarrow \textit{next}(\textit{rand}, (\textit{rand}, \textit{ag3}, \text{tell}, 1)) \\
\mathcal{S}_6 &: \ \Phi \rightarrow \textit{next}(\textit{rand}, (\textit{rand}, \textit{ag3}, \text{tell}, 2))
\end{aligned}$$

Strategies $\mathcal{S}_1$ and $\mathcal{S}_2$ are for *ag1*; $\mathcal{S}_3$ and $\mathcal{S}_4$ are for *ag2*; $\mathcal{S}_5$ and $\mathcal{S}_6$ are for *ag3*.

Next we look at the possible outcomes of the protocol. The net effect of the protocol is the creation of a persistent social fact which describes that the winner must buy the lot at the agreed price and the auctioneer must sell it to him; we do this with a 'COMMIT' fact. We will not discuss the details of how this buying phase is brought about; we assume that it entails the performance of some legally binding actions which transfer ownership of the auctioned lot as well as the required funds. The commitment exists only for the bidder, the auctioneer is not explicitly constrained by it; however, we can assume that the bidder is thereby empowered to force the auctioneer to complete its part of the deal. We are missing some social facts to encode the required institutional constraints, but this is not the focus of this work; it suffices to say that the commitment to buy is itself legally binding and a guarantee that the protocol outcome results in this commitment is good enough for the participating agents. Each outcome describes the final value of the persistent social facts function p_i, given that its initial value is p_i^0.

$$\begin{aligned}
\mathcal{O}_1 : p_{\textit{ag1}} &= [\text{'COMMIT } \textit{ag1}, (\text{buy,lot,}\textit{ag3}\text{,3+2/3)'} \mapsto_{\text{T}}] p^0_{\textit{ag1}} \ \wedge \ p_{\textit{ag2}} = p^0_{\textit{ag2}} \\
\mathcal{O}_2 : p_{\textit{ag2}} &= [\text{'COMMIT } \textit{ag2}, (\text{buy,lot,}\textit{ag3}\text{,3+2/3)'} \mapsto_{\text{T}}] p^0_{\textit{ag2}} \ \wedge \ p_{\textit{ag1}} = p^0_{\textit{ag1}} \\
\mathcal{O}_3 : p_{\textit{ag1}} &= [\text{'COMMIT } \textit{ag1}, (\text{buy,lot,}\textit{ag3}\text{,3)'} \mapsto_{\text{T}}] p^0_{\textit{ag1}} \ \wedge \ p_{\textit{ag2}} = p^0_{\textit{ag2}} \\
\mathcal{O}_4 : p_{\textit{ag2}} &= [\text{'COMMIT } \textit{ag2}, (\text{buy,lot,}\textit{ag3}\text{,3)'} \mapsto_{\text{T}}] p^0_{\textit{ag2}} \ \wedge \ p_{\textit{ag1}} = p^0_{\textit{ag1}}
\end{aligned}$$

If f is a function, then $[x \mapsto y]f$ denotes the function that maps x to y, but behaves like f for any other argument. Now we calculate the formula $\mathcal{M}$ which is a formal specification for the desired mechanism. It describes, for every possible combination of agent strategies, the resulting outcome. The $\mathcal{U}$ (until) operator has been used to specify that Ψ will be false initially, and will remain false until both it is true and the specified protocol outcome is also true; this also guarantees eventual termination of the protocol. Of the six final conjuncts (in $\mathcal{M}$ below), the first states that if $ag1$ has declared high (strategy $\mathcal{S}_1$) and his opponent $ag2$ has declared high (strategy $\mathcal{S}_3$) and

$rand$ has sent the number '1' (strategy $\mathcal{S}_5$) then $ag1$ will get the lot and pay $3\frac{2}{3}$ for it (outcome $\mathcal{O}_1$); the other properties are similar. Since individuals are rational and risk neutral the opponent declares high with probability $\frac{1}{2}$. We also know that $rand$ replies with the number '1' with probability $\frac{1}{2}$. Thus, given that $ag1$ bids high, this first conjunct gives an outcome (he wins and pays $3\frac{2}{3}$) which occurs with probability $\frac{1}{4}$. The second conjunct gives an outcome ($\mathcal{O}_2$, he loses and his opponent wins and pays $3\frac{2}{3}$) which occurs with probability $\frac{1}{4}$. The fifth conjunct gives an outcome ($\mathcal{O}_3$, he wins and pays $3\frac{2}{3}$) which occurs with probability $\frac{1}{2}$. These three properties cover all the possible outcomes when $ag1$ bids high i.e. in total he wins with probability $\frac{3}{4}$ and pays $3\frac{2}{3}$ whenever he wins. On the other hand, if $ag1$ bids low, outcome $\mathcal{O}_3$, $\mathcal{O}_4$ or $\mathcal{O}_2$ will occur, giving a probability $\frac{1}{4}$ to win and pay 3. These results are in line with the discussion in section 5 and guarantee incentive compatibility and individual rationality for both $ag1$ and $ag2$ (the protocol is symmetric) and the optimal price for the seller. So our task now is to specify a protocol which has property $\mathcal{M}$.

$$\mathcal{M} : \forall\, p^0_{ag1},\, p^0_{ag2},\, p^0_{ag3} : \begin{bmatrix} p^0_{ag1} = p_{ag1} \wedge \\ p^0_{ag2} = p_{ag2} \wedge \\ p^0_{ag3} = p_{ag3} \end{bmatrix} \rightarrow \begin{bmatrix} \Phi \wedge \mathcal{S}_1 \wedge \mathcal{S}_3 \wedge \mathcal{S}_5 & \rightarrow \neg\Psi\, \mathcal{U}(\Psi \wedge \mathcal{O}_1) \wedge \\ \Phi \wedge \mathcal{S}_1 \wedge \mathcal{S}_3 \wedge \mathcal{S}_6 & \rightarrow \neg\Psi\, \mathcal{U}(\Psi \wedge \mathcal{O}_2) \wedge \\ \Phi \wedge \mathcal{S}_2 \wedge \mathcal{S}_4 \wedge \mathcal{S}_5 & \rightarrow \neg\Psi\, \mathcal{U}(\Psi \wedge \mathcal{O}_3) \wedge \\ \Phi \wedge \mathcal{S}_2 \wedge \mathcal{S}_4 \wedge \mathcal{S}_6 & \rightarrow \neg\Psi\, \mathcal{U}(\Psi \wedge \mathcal{O}_4) \wedge \\ \Phi \wedge \mathcal{S}_1 \wedge \mathcal{S}_4 & \rightarrow \neg\Psi\, \mathcal{U}(\Psi \wedge \mathcal{O}_1) \wedge \\ \Phi \wedge \mathcal{S}_2 \wedge \mathcal{S}_3 & \rightarrow \neg\Psi\, \mathcal{U}(\Psi \wedge \mathcal{O}_2) \end{bmatrix}$$

7 Protocol Specification

Our scenario has few social facts, these include permission, obligation, commitment and social variables to describe roles and other aspects of the conversation; the semantics for these is specified below, with implicit quantification for all i, j, *act*, *content*.

$$\begin{aligned} &[\![\text{'PERMIT } i, \text{(wait)'}, i]\!]_f = \bigcirc \neg[\alpha_{j,i}\!\leqslant] \\ &[\![\text{'OBLIGE } i \text{ } (j, \mathit{act}, \mathit{content})\text{'}, i]\!]_f = \bigcirc\, (\alpha_{j,i}\!\leqslant\!(i, j, \mathit{act}, \mathit{content}) \vee \neg[\alpha_{j,i}\!\leqslant]) \\ &\qquad\qquad \wedge \Diamond\, \alpha_{j,i}\!\leqslant\!(i, j, \mathit{act}, \mathit{content}) \\ &[\![\text{'OBLIGE } i \text{ OR } (j_1, \mathit{act}_1, \mathit{content}_1) + (j_2, \mathit{act}_2, \mathit{content}_2)\text{'}, i]\!]_f \\ &= \bigcirc\, (\alpha_{j,i}\!\leqslant\!(i, j_1, \mathit{act}_1, \mathit{content}_1) \vee \alpha_{j,i}\!\leqslant\!(i, j_2, \mathit{act}_2, \mathit{content}_2) \vee \neg[\alpha_{j,i}\!\leqslant]) \\ &\quad \wedge \Diamond\, (\alpha_{j,i}\!\leqslant\!(i, j_1, \mathit{act}_1, \mathit{content}_1) \vee \alpha_{j,i}\!\leqslant\!(i, j_2, \mathit{act}_2, \mathit{content}_2)) \end{aligned} \tag{1}$$

Note, it is not intended that permissions and obligations here are related to deontic logic operators in any way and permission is not the dual of obligation. The semantic function maps a permission to wait to the set of models where the agent does nothing next ($\bigcirc$ is the temporal operator for next and $\neg[\alpha_{j,i}\!\leqslant]$ means that i sends no communication to j). An obligation for an agent i maps to a set of models where agent i either does the action or does nothing in the next state; and eventually i does do the action. Thus our obligation can be interpreted as "do this and do nothing else until it is done". We have opted for this in preference to a simple permission because we want to guarantee that the auction will terminate. We have not given the semantics of 'COMMIT' as we have omitted the buying phase (see the discussion in the previous section). All other social facts such as roles are trivially satisfied in all models.

The initial condition Φ for true social facts is shown in equation 2, all other contingent social facts map to false. It is a conjunction of assignments explaining the initial obligations, permissions and roles. For example, the value of c_{ag1} maps the social fact 'bidder1=ag1' to true. Therefore we assume a prior initiation phase where bidders have already made a decision to enter, this decision being based on the protocol properties we are about to prove; especially the property of individual rationality.

$$\begin{array}{l} c_{ag1}[\text{OBLIGE } ag1 \text{ OR } (ag3,\text{tell,H}) + (ag3,\text{tell,L})] \wedge \\ c_{ag2}[\text{OBLIGE } ag2 \text{ OR } (ag3,\text{tell,H}) + (ag3,\text{tell,L})] \wedge \\ c_{ag3}[\text{PERMIT } ag3 \text{ (wait)}] \wedge c_{rand}[\text{PERMIT } rand \text{ (wait)}] \wedge \\ \forall i \in \{ag1,ag2,ag3\} : \quad \left[\begin{array}{l} c_i[\text{protocol=simple\_auction}] \wedge c_i[\text{auct=ag3}] \wedge \\ c_i[\text{bidder1=ag1}] \wedge c_i[\text{bidder2=ag2}] \wedge \\ c_i[\text{lot=goods\_description}] \wedge c_i[\text{item=forsale}] \end{array}\right] \end{array} \tag{2}$$

The final condition Ψ, which remains false until termination of the protocol is:

$$\Psi : \forall i \in \{ag1,ag2,ag3,rand\} : c_i[\text{protocol=nil}] \tag{3}$$

The protocol is specified in figure 1 in an ACL specification language; we have shown in previous work [7] how this specification language can be given a denotational semantics, mapping it to the ACL semantics function $[\![-]\!]_c$ described in section 3.3.

The auction opens with social facts for *bid1* and *bid2* unassigned. Therefore the bidders are obliged to send a *tell* message declaring their bid, high or low. When both bidders have declared their valuation, if they are equal, the auctioneer is obliged to request an integer from *rand*. The semantics of this request creates a social fact which is an expressed desire for a number. Following from this expressed desire the agent *rand* is obliged to reply with a *tell* message with content 1 or 2. Now if the number was 1 and the bid was high the auctioneer awards the sale to *bidder1* for '3+2/3' and so on. If both bids are different the auctioneer awards the item to the highest bidder without the intervention of *rand*. Finally, in the speech act semantics for *award* we specify that the winning bidder is committed to buy the 'lot' at the agreed price.

8 Proof of Protocol Properties

We described the properties of the mechanism in section 6, now we show that the protocol we have specified has these properties. To prove property $\mathcal{M}$ for our protocol we must show that formula $\mathcal{M}$ holds for any system of compliant agents executing the protocol. $\mathcal{M}$ must hold over all compliant computations of the multi-agent system S_E (see section 3.4) where Φ (equation 2) is the initial condition for S_E. We prove this by constructing a state diagram for the multi-agent system S_E (see figure 2). This diagram starts at the initial social state and thereafter takes all compliant transitions and shows the resulting social states. There are two temporal operators in our semantics for social facts (see equation 1): $\bigcirc$ and $\Diamond$, in taking our compliant transitions in the diagram we have respected the $\bigcirc$ part but ignored $\Diamond$. Thus the possible observable computations

```
converse-function
[simple_auction]
 if #bid1=nil
  then {OBLIGE #bidder1 OR (#auct,tell,H)+(#auct,tell,L)}
  else {PERMIT #bidder1 (wait)};
 if #bid2=nil
  then {OBLIGE #bidder2 OR (#auct,tell,H)+(#auct,tell,L)}
  else {PERMIT #bidder2 (wait)};
 if E-desire #auct number and #number=nil
  then {OBLIGE #rand OR (#auct,tell,1)+(#auct,tell,2)}
  else {PERMIT #rand (wait)};
 if #bid2=#bid1 and #bid1!=nil
  then {if #number=nil and not E-desire #auct number
         then {OBLIGE #auct (#rand,request,number)}
         else
         {if #number=1
           then {if #bid1=H
                  then {OBLIGE #auct (#bidder1,award,3+2/3)}
                  else {OBLIGE #auct (#bidder1,award,3)};}
           else {if #bid1=H
                  then {OBLIGE #auct (#bidder2,award,3+2/3)}
                  else {OBLIGE #auct (#bidder2,award,3)};}}}
  else {if #bid1=H and #bid2=L
         then {OBLIGE #auct (#bidder1,award,3+2/3)}
         else {if #bid1!=nil
                then {OBLIGE #auct (#bidder2,award,3+2/3)}
                else {if #item=sold
                       then {OBLIGE #auct (all,announce,nil)};
                       else {PERMIT #auct (wait)};}}};
protocol-semantics
 [simple_auction]
  [tell]if #bidder1=S then {bid1=C};if #bidder2=S then {bid2=C};
        if #rand=S then {number=C}
  [award]item=sold
speech-act-semantics
 [announce] protocol=C
 [request]  E-desire S C
 [award]    COMMIT R (buy,lot,S,C)
```

Fig. 1. ACL Specification for Auction Protocol.

of a compliant system of agents using the protocol will be a subset of (or equal to) the possible paths in the diagram; there may be paths in the diagram which violate the $\Diamond$ part of the social fact semantics. To avoid clutter in the diagram we do not provide a list of all the social facts true at each state, just the important ones; we also omit the messages themselves, these are obvious by inspection of the source and destination states of edges in the diagram; the terminal states are not included either, but there are six and they are each reached from one of the final states in the diagram by means of the final announcement. All transitions shown are τ_E transitions (involving message passing); τ_I transitions have been omitted but exist for all states and connect a state to itself.

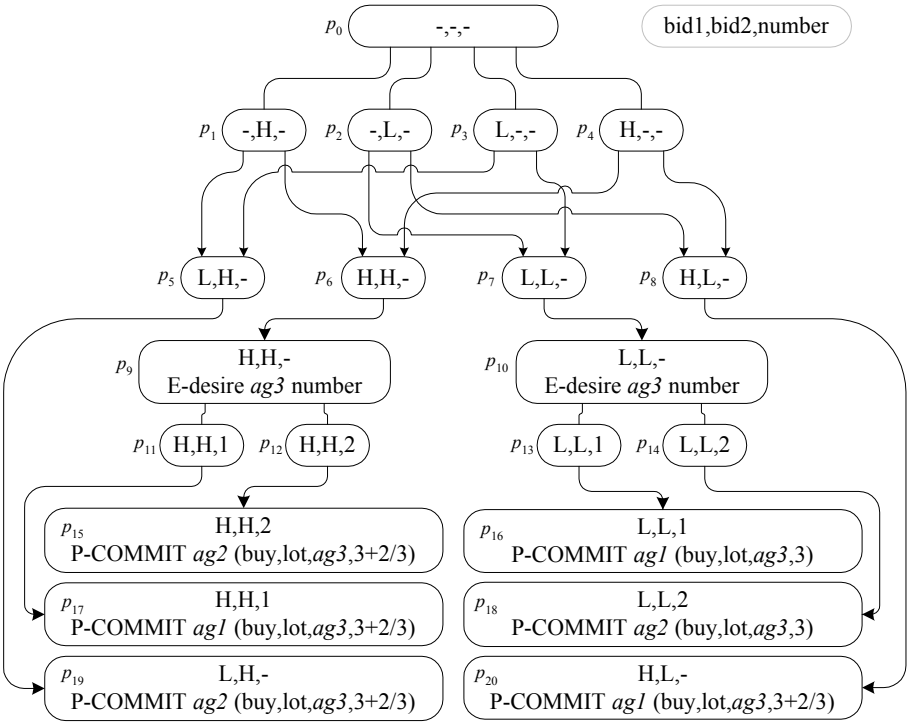

Fig. 2. State Transition Diagram for S_E (Also Protocol Diagram for Auction).

We can take each conjunct in $\mathcal{M}$ in turn to prove its validity, starting with the first:

$$\begin{aligned}
\varphi_1: \quad & [\Phi \wedge \mathcal{S}_1 \wedge \mathcal{S}_3 \wedge \mathcal{S}_5] \rightarrow \ [\neg\Psi\, \mathcal{U} (\Psi \wedge \mathcal{O}_1)] \\
= \ & \neg[\Phi \wedge \mathcal{S}_1 \wedge \mathcal{S}_3 \wedge \mathcal{S}_5] \vee \ [\neg\Psi\, \mathcal{U} (\Psi \wedge \mathcal{O}_1)] \\
\neg\varphi_1: \quad & [\Phi \wedge \mathcal{S}_1 \wedge \mathcal{S}_3 \wedge \mathcal{S}_5] \wedge \neg[\neg\Psi\, \mathcal{U} (\Psi \wedge \mathcal{O}_1)] \\
= \ & \Phi \wedge \mathcal{S}_1 \wedge \mathcal{S}_3 \wedge \mathcal{S}_5 \ \wedge \ [\neg(\Psi \wedge \mathcal{O}_1)\, \mathcal{W} (\Psi \wedge \neg(\Psi \wedge \mathcal{O}_1))] \\
= \ & \Phi \wedge \mathcal{S}_1 \wedge \mathcal{S}_3 \wedge \mathcal{S}_5 \ \wedge \ [(\neg\Psi \vee \neg\mathcal{O}_1)\, \mathcal{W} (\Psi \wedge \neg\mathcal{O}_1)]
\end{aligned}$$

Note that $\neg(p \mathcal{U} q)$ can be rewritten as $\neg q\, \mathcal{W}(\neg p \wedge \neg q)$ where $\mathcal{W}$ is the *waiting for* operator [8]. Now we can prove the validity of φ_1 by failing to find a path in the diagram on which $\neg\varphi_1$ is true. We are searching for an infinite path on which $\neg\varphi_1$ holds, starting at the initial node. Since there are no loops in our diagram, the path must progress to a terminal node on a finite path through the diagram and thereafter continue on an infinite path. A search of the graph reveals that a path which satisfies $\Phi \wedge \mathcal{S}_1 \wedge \mathcal{S}_3 \wedge \mathcal{S}_5$ must come to node p_{17}. Now the final conjunct requires that $(\neg\Psi \vee \neg\mathcal{O}_1)$ remains true, waiting for $(\Psi \wedge \neg\mathcal{O}_1)$ to be true. On the two paths up to p_{17} we do have $(\neg\Psi \vee \neg\mathcal{O}_1)$ true and $(\Psi \wedge \neg\mathcal{O}_1)$ is false; however, if we go on from p_{17} to the terminal protocol state (not shown in the diagram) we will have both Ψ and $\mathcal{O}_1$ true, but still $(\Psi \wedge \neg\mathcal{O}_1)$ is false so this path cannot satisfy $\neg\varphi_1$. The validity of the remaining φ_i formulas can be shown in the same way. Since the protocol has a finite number of states, the procedure could

be automated with a model checker. This proves that our auction protocol satisfies $\mathcal{M}$ and so it has all the properties we desire. The type of enforcement we would require, to ensure that the auction properties hold, is a sentinel agent who monitors all transactions. In the absence of this, the auctioneer could violate the protocol constraints and the bidders could not know. If we want self policing we would have to redesign the protocol so that all speech acts are broadcast and all agents (in this case we could not use a sealed bid auction); then agents know what the other's bid is, what number is returned by *rand* and therefore the action the auctioneer should take. This protocol is still not completely deception free however, it does not prevent bidder collusion with the auctioneer or rings [12]. For example the two bidders could privately value the item high and declare low, one of them paying less than the valuation for the item and then giving the other agent some side payment. Such deceptions happen in real (human) auctions also, in fact we consider an open agent society to be quite like a human society in terms of the extent to which we can constrain the behaviour of individuals.

9 Conclusions and Future Work

Given that there are already numerous examples in the literature of winner determination algorithms for more complex and more general classes of auctions than our example [13, 4], what have we achieved? We have made the auctioneer's winner determination algorithm public so that designers of bidding agents can inspect it. We have provided a procedure by which they can verify the properties of the protocol. If the society is policed in some way (section 4.1) we can guarantee that the protocol properties hold for a system of agents using the protocol. Our formal framework makes this policing possible since it allows us to identify rogue agents at run time. Therefore, in our example, we can guarantee that the protocol is symmetric (it doesn't favour one bidder) and that a bidder's best strategy is to bid truthfully. The protocol designer typically designs a protocol to induce agents to use some dominant strategy. It is therefore important that the specification is public so participants know what the dominant strategies are. For example if we want bidders to bid truthfully in our auction they must be assured of incentive compatibility. In general our framework makes it possible to guarantee to an agent owner that an agent's participation in a system does not result in any disadvantageous deal for it. This will increase the range of e-commerce applications in which an agent owner will be willing to delegate a task to an agent.

The example protocol we have presented is extremely simple. Catering for a more general class of auctions (multiple bidders, prices and items etc.) will require extending the ACL specification language to allow more mathematical functions to be used in protocol specification. In principle it should be possible to publicly specify any mechanism which can be programmed into an agent, since any program can be given a formal semantics. Future work will seek a richer representation of the social context, to include empowerment, norms and institutions [14]. There is also a need for a comprehensive formalisation of desirable properties of mechanisms; it should then be possible to design automatic tools which could analyse a protocol specification and determine which of the properties discussed in section 2 hold for it. Our work is a significant step in this direction because our method of specifying the protocol rules uses a specification

language which has been given a denotational semantics; this means that there is a well defined procedural interpretation for agent actions and hence tools for automatic verification are straightforward to implement.

Acknowledgements

We are grateful to Dr. Mehdi Dastani for providing the ideas which led to this paper. We also acknowledge the constructive criticism given by the anonymous reviewers.

Bibliography

[1] Axelrod, R. (1984). *The evolution of cooperation*. Basic Books, New York.

[2] Béjar, J. and Cortés, U. (2001). Agent strategies on dpb auction tournaments. In *Dignum, F., Cortés, U., Agent-Mediated Electronic Commerce III. Springer-Verlag.*

[3] Binmore, K. (1992). *Fun and Games: A text on Game theory*. D.C. Heath and Company, Lexington, Massachusetts.

[4] Easwaran, A. M. and Pitt, J. (2000). An agent service brokering algorithm for winner determination in combinatorial auctions. In *European Conference on Artificial Intelligence*. (ECAI), Berlin, Germany.

[5] Eijk, R. M. v. (2000). *Programming Languages for Agent Communication*. PhD thesis, Department of Information and Computing Sciences, Utrecht University.

[6] Esteva, M., Rodriguez, J., Sierra, C., and Garcia, P. (2001). On the formal specification of electronic institutions. In *LNAI 1991*, pages 126–147. Springer.

[7] Guerin, F. and Pitt, J. (2001). Denotational semantics for agent communication languages. In *Autonomous Agents 2001, Montreal*, pages 497–504. ACM Press.

[8] Manna, Z. and Pnueli, A. (1995). *Temporal Verification of Reactive Systems (Safety), vol. 2*. Springer-Verlag, New York, Inc.

[9] Parkes, D. C. (2000). Optimal auction design for agents with hard valuation problems. In *Agent Mediated Electronic Commerce II: Towards Next-Generation Agent-Based Electronic Commerce Systems (LNAI 1788), Eds: Moukas et al., Springer.*

[10] Pradella, M. and Colombetti, M. (2001). A formal description of a practical agent for e-commerce. In *Dignum, F., Cortés, U., Agent-Mediated Electronic Commerce III. Springer-Verlag.*

[11] Rosenschein, J. and Zlotkin, G. (1994). *Rules of Encounter*. MIT Press.

[12] Sandholm, T. (1996). *Negotiation among Self-Interested Computationally Limited Agents*. PhD thesis, University of Massachusetts at Amherst, C.S. Department.

[13] Sandholm, T., Suri, S., Gilpin, A., and Levine, D. (2001). Cabob: A fast optimal algorithm for combinatorial auctions. In *International Joint Conference on Artificial Intelligence*, pages 1101–1108. (IJCAI), Seattle, WA.

[14] Singh, M. (1999). An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7:97–113.

[15] Singh, M. (2000). A social semantics for agent communication languages. In *IJCAI Workshop on Agent Communication Languages, Springer-Verlag, Berlin.*

[16] Wooldridge, M. (2000). Semantic issues in the verification of agent communication languages. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(1):9–31.