Protocols and Intentional Specifications of Multi-Party Agent Conversations for Brokerage and Auctions

Jeremy Pitt
Dept. of Electrical & Electronic Engineering
Imperial College of Science, Technology & Medicine
London, SW7 2BT, UK
j.pitt@ic.ac.uk

Frank Guerin
Dept. of Electrical & Electronic Engineering
Imperial College of Science, Technology & Medicine
London, SW7 2BT, UK
f.guerin@ic.ac.uk

Chris Stergiou
Dept. of Electrical & Electronic Engineering
Imperial College of Science, Technology & Medicine
London, SW7 2BT, UK
c.stergiou@ic.ac.uk

ABSTRACT

Conversations involving three or more agents often occur in multi-agent systems, for example in brokering and auctions. For developing agents in open systems, it is important that the interactions in such conversations have a precise and unambiguous meaning. We address this issue by generalising a protocol-based semantic framework for expressing the semantics of Agent Communication Languages. The generalisations involve exploiting mechanistic aspects of the interaction (conversation identifiers), greater flexibility in the space of possible replies, and a richer representation of protocol states. We define intentional specifications for some brokerage and auction protocols, including event-based clocks to determine the ordering of events. We conclude that this approach to specifying multi-party protocols leads to clearer interfaces for open systems and easier re-use, with a potentially significant impact on standardisation efforts.

1. INTRODUCTION

Conversation policies and interaction protocols have proved useful and indeed almost essential for high-level interoperability between heterogeneous agents in multi-agent systems (MAS). However, they have primarily been based on one-to-one conversations, i.e. dialogues between only two agents. It is a feature of many MAS, though, that there is some 'well known' agent (cf. *well known ports* in TCP/IP networks) that provides generic facilities to all other agents, for example directory and management services.

For example, the KQML language specification includes a Facilitator agent [3]. Facilitators support multi-agent and third-party conversations in some commonly recurring patterns of interaction between three agents. In some cases, there is even indirection, as one agent may not be aware of who one of the others is in the conversation.

For developing agents in open systems, it is important that the interactions in such conversations have a precise and unambiguous meaning. This paper addresses the issue by generalising the protocol-based, semantic framework of Pitt and Mamdani [6; 5], developed for describing the semantics of Agent Communication Languuges (ACLs) at different levels of abstraction. These generalisations enable us to provide formal specifications of the interaction patterns described in [3], and also for auction protocols.

Section 2 reviews the KQML brokerage protocols and the nature of the semantic problem, and reviews the general semantic framework which will be used to frame a solution. Section 3 demonstrates the solution, which is based on exploiting conversation identifiers and a richer space of possible replies. Section 4 gives a further generalisation which caters for auction protocols. In both cases we give an intentional specification of expected agent behaviour. Section 5 summarises the work, and argues that interpreting speech acts in context is preferable to using complex speech acts. We conclude that the emphasis on design of multi-party protocols can lead to 'standardised' interfaces for open systems, easier re-use, and can expose unexpected problems.

2. BACKGROUND & MOTIVATION

2.1 Multi-Party Conversations & Protocols

There is a definite requirement for multi-party conversations in MAS applications where brokerage and/or auctions are required. in KQML a special class of agent called facilitators was introduced to perform various useful communication services, in particular mediation or brokerage services. In [3], four interaction patterns based on these services were described, for recruitment, brokerage, recommendation and subscription (clockwise from top right in Figure 1).

As with the well-known contract-net protocol, the value of these protocols was that, given the frequency with which they occurred in different applications, the specifications could be simply re-used. However, the problem with understanding and applying these diagrams is that in [3], the semantics of KQML was an open issue. This meant some difficulty in interpretation. For example, in the recommend protocol, agent A was supposed to ask F if there was an agent willing to provide a certain service (e.g. $\mathsf{ask}(X)$), and F would reply "once it learns that B is willing to accept $\mathsf{ask}(X)$ performatives" [3]. But this means that agent A may have to block until B advertises, whereas in fact it

Agents 2000 Barcelona Spain

would be more useful for A to know directly that no agent has advertised the service it requires.

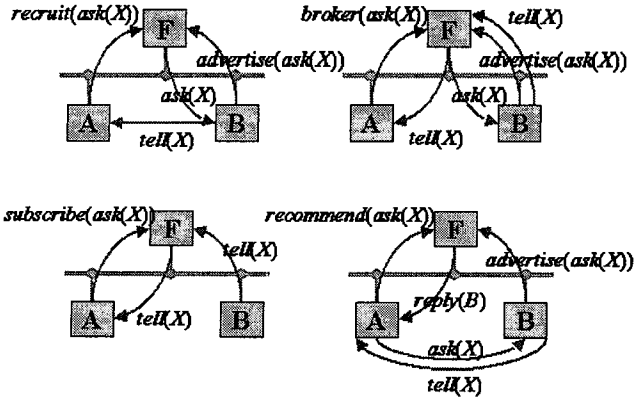

Figure 1: KQML communication facilitation services

This problem has largely been addressed in KQML by the use of Coloured Petri Nets [2], but it remains a problem for the FIPA ACL semantics [4]. In a series of papers submitted to FIPA, an attempt has been made to express the semantics of the broker communicative act in the SL logic of the FIPA semantics [9; 8]. The experience has been that it is very hard to do, understand, verify and apply.

Nevertheless, these protocols have been implemented and widely used in a variety of applications. Probably, a number of specific solutions have been developed but it is unlikely that such systems could then interoperate. What we are trying to achieve with this paper is to show how such protocols can be specified in a general semantic framework. This framework is briefly reviewed in the next section.

2.2 The General Semantic Framework

The protocol-based semantic framework has been introduced in [6]. The main idea in the original work was to separate out the 'external', action-level semantics (i.e. observed speech acts) from the intentional semantics (agent-internal reasons for and acting and replying) and the content-level semantics (i.e. the meaning of the content actually communicated). This core idea reamins but we extend the framework slightly to handle multi-party protocols.

An ACL is a 3-tuple $< Perf, Prot, reply >$ where $Perf$ is a set of perfomative names, $Prot$ is a set of protocol names, and $reply$ is a partial function given by:

$$reply : Perf \times Prot \times \mathrm{N}^{+} \longmapsto P(Perf \times Prot)$$

where N^{+} is the domain of positive integers. The reply function is then defined for each distinct state of each protocol, identified by a unique (for each protocol) integer. This gives for each speech act, 'performed' in the context of a conversation being conducted according to a specific protocol, what performatives in which protocols are acceptable replies. The $reply$ function therefore specifies a finite state diagram for each protocol named in Prot. (Note also that in $Perf$ we include the null performative which is a 'do nothing' (no reply) performative (cf. 'silence' in [7])).

An agent s communicates with (and communicates information to) an agent r via a speech act. This (possibly infinite) set is denoted by $speech\_acts$, a single member sa of which is represented by:

$$sa =< s, \mathsf{perf}(r, (C, L, O, cp, ci, t_s)) >$$

This is saying that s does (communicates with) performative perf with content C in language L using ontology O in the context of protocol (conversation policy) cp as part of a conversation identified by ci at time of sending t_s. The notation $sa.perf$ denotes the performative of a speech act, and so on.

The meaning of such a speech act sa from agent s to agent r is then given by:

$$[\![< s, \mathsf{perf}(r, (C, L, O, cp, i, t_s)) >]\!] = \mathcal{I}_r < r, rep\_sa > \text{ s.t.}$$
$$(rep\_sa.perf, rep\_sa.prot) \in reply(\mathsf{perf}, cp, conv_r(i))$$

This means that, in this framework, at the observable action level, the meaning of a speech act is the intention to give a reply. Note that this defines only the meaning of a speech act at the action level. This is therefore an external semantics, i.e. from the perspective of an observer that cannot 'see' the agent internals. It may well be that two speech acts get the same apparent meaning (for example, whenever the intended reply is null). However, we contend that from the observer's perspective that is exactly right: all the observer sees is the actions. If two different actions don't entail any response then the actions mean the same thing (and in the absence of any observable change of state, are ostensibly meaningless (sic)) – at the action level.

To fully characterise the intended semantics, three further functions are required, which are specified relative to each agent a, and state what that agent does with a message, not how it does it. The three functions in [6] were (1) a procedure for computing the change in an agent's information state from the content of an incoming message; (2) a procedure for selecting a performative from a set of performatives (valid replies), and (3) a function $conv$ which mapped a conversation identifier onto the current state of the protocol. For these functions, we specify intentional (logical) descriptions of the reasons for and reactions to a speech act. These serve as reference implementation models that agent developers could use to implement the appropriate internal and external behaviours for their agents. Furthermore, where the import of the the content level meaning was required, further speciifcations could be supplied, and this is dependent upon the application.

Therefore, the protocol-based semantic framework supports a general methodology for designing an ACL for a particular application [5]:

- Generalization at the action level: additional performatives and protocols can be introduced to create a new ACL and new patterns of interaction;
- Specialization at the intentional level: a reference implementation model, possibly referring to agents' beliefs, desires and intentions, could be specified to give intended behavioural meanings for performatives in the context of the protocols;
- Instantiation at the content level: the specific decision-making functionality for deciding which reply from a set of allowed replies can also be specified. For example, the same protocol may be used in quite different application domains and the decision making may be dependent on a number of different factors.

We will now show how the brokerage protocols of [3] can be specified at the action level, and give logical specifications to complete the meaning at the intentional level.

3. A SEMANTICS FOR BROKERAGE

3.1 Conversation Identifiers

Conversation identifiers in ACLs are used to identify a dialogue between two agents, so that a message in one conversation is not confused with message in another, simultaneous, conversation. In the FIPA specifications, the conversation identifier is a parameter of the ACL message, but it is not part of the semantics. It is noted that there are parts of the specification that fall outside the scope of the semantics, and it is arguably this omission that makes the logical specification of the broker performative harder than it needs to be, because there is no 'handle' in the semantics for referring to a conversation as a 'first-class object'.

Our approach brings the conversation identifiers into the semantics, but these need to be unique. One way of guaranteeing uniqueness is to use the FIPA naming system for Globally Unique Identifiers for agent names and a timestamp generated from the system clock. Modulo any operating system vagaries, a unique identifier is then generated for each conversation in which an agent participates [10].

An alternative mechanism for generating unique conversation identifiers is to use a 2-tuple, and for each participant to supply one component: we use an integer.

Each agent maintains a count of the number of conversations it has engaged in, and increments this by one for each new conversation. Then, it assigns the current count for its first speech act in the conversation (see Figure 2).

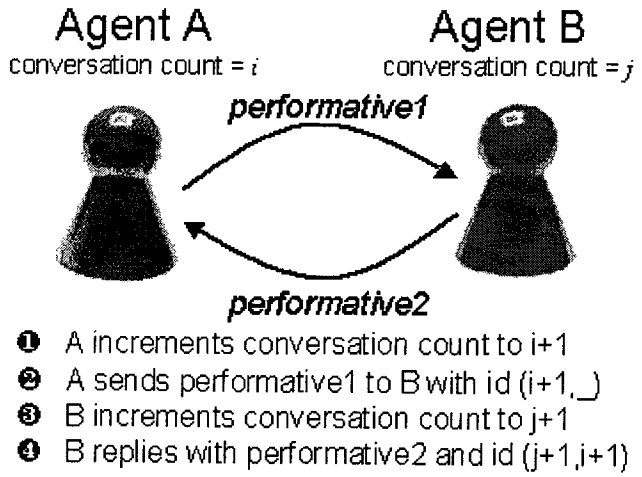

Figure 2: Communication with conversation identifiers

If both agents in a conversation do this, then the combination of protocol state and the conversation identifier uniquely identifes the conversation and which message has been sent in reply to which other, i.e. there is no need to have separate `:reply-with` and `:in-reply-to` message parameters in the ACL. Where no reply is expected or required, the receiving agent need take no action about replying, but still labels the conversation for later reference.

The advantage of having one component of the 2-tuple conversation identifier provided by each party is that each one can then label its own component with additional information, and return the other party's information intact. The conversation identifier can be treated as a term (in the Prolog sense) and can contain structured information. The functor of the term is the uniquely-assigned integer, and the arguments can be the extra information.

This then means that the conversation identifiers can be treated like "Our Reference" and "Your Reference" labels in human information transactions (e.g. memo passing). By convention, one uses one's own system for "our reference", and respects whatever system is used by the other party. The advantage for agent communication is the knowledge of how to interpret the reference in context: for example, in the KQML-style communication [3] of Figure 3, the interpretation of the reference $j(A,i)$ is for agent B to send the reply to the content of the broker (speech act) received from the Faciliator F to agent A quoting its reference i. (Note also by convention in Figure 3, we put the sender's reference first and the receiver's reference second.) Then, the conversation identifier is constant but the additional information can be variable throughout the lifetime of a single conversation.

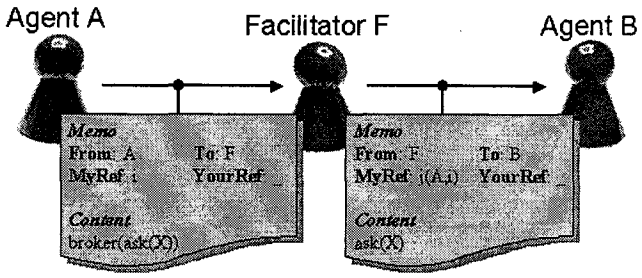

Figure 3: Parameterised conversation identifiers

We define the following function cid_a for each agent a:

$$cid_a : \mathbb{I} \times AgentIdentifier \longrightarrow \mathbb{I}$$

This gives, for each agent a, for a conversation identifier and the name used by a to identify some other agent, the identifier used by that agent to identify the same conversation.

3.2 Protocols & Intentional Specifications

The KQML brokerage protocols can each be analysed as a composite of two or three separate one-to-one conversations, which can be intrinsically linked by the *reply* function. In this section, these conversations will first be specified at the action level as finite state diagrams, with the underlying meaning given by intentional specifications. The formal language for the specification is a multi-modal dynamic action logic using the 'triggers and tropisms' style of [5], i.e. specifying the reason for doing and responding to speech acts. Formulas like $[a, A]p$ intuitively state that after agent a does action A, then an agent which 'observed' A and has this formula in its belief state should endeavour to make p true.

Note that after the first speech act in a protocol, we assume that the receiver assigns its component of the conversation identifier, and that after that DONE($< a, \mathsf{perf}(\ldots, (i,j)) >$) is true, where (i,j) is the conversation identifier.

The finite state diagrams for the various parts of the recruit and broker performative/protocols are illustrated in Figure 4. The finite state diagrams for subscribe and recommend are simple variations on a simple question/answer pair. Intentional specifications of the meaning of the speech acts in the context of these protocols, together with an English paraphrasal, are given in Table 1 (note we use ad. as an abbreviation for advertise).

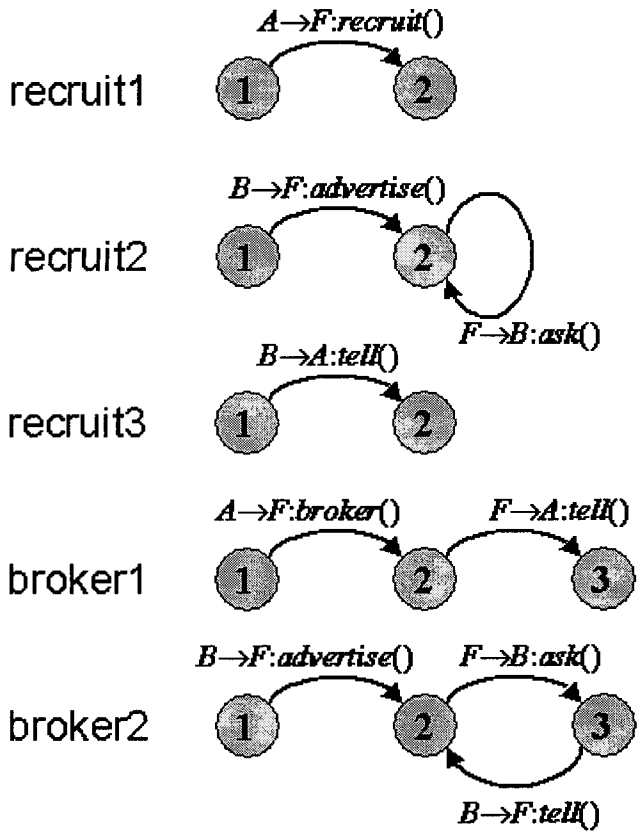

Figure 4: Finite State Diagrams for KQML Brokerage

3.3 Extending the Framework

The first observation to make is that a conversation identifier is no longer a single unique identifier, as specified in [6], but is a pair, one element of which is supplied by each party to the conversation. The function of the identifiers is the same as before, but the fact that they can be parameterised allows us to provide additional information. This information can be interpreted (in the context of a protocol) to forward to third parties and so on, giving a precise formal specification of the KQML brokerage protocols.

The second observation is that under the previous semantics, an agent either replied with a speech act in the same protocol, or, if the conversation was over, performed the null speech act. The above logical specifications are correct with respect to that semantics; however, it is not quite capturing the notion that although a speech act in this conversation is not expected (or allowed), another speech act in a new conversation is intended. Therefore the range of the *reply* function is a set of pairs of speech act and protocol.

The third observation to make is that using these generalisations in conjunction with the intentional specifications, we can be more precise about the order of events and what actions the Facilitator should take. The value of such specifications is that it is now clear what the order of actions is expected to be, and under what circumstances certain actions are expected. Note that the protocols remain normative in that they specify what actions can (must) be done, but the intentional specifications give declarative specifications of what decisions should be taken, but do not specify *how* this decision making should be implemented. In this sense the intentional specification is informative, but any agent hoping to work in a system based on this specification of the protocols is expected to comply with these intentions, and in this sense the specification is providing a reference implementation model for an agent implementer.

However, even now there are certain elements which remain underspecified: for example, actions to recover from errors (for example, in lines 1–3, if there was not an agent who

1	$[A, \mathsf{recruit}(F, \mathsf{ask}(X), \mathit{recruit1}, (i, \_))]$
2	$\exists b, j, k.\mathrm{DONE}(< b, \mathsf{ad}.(F, \mathsf{ask}(X), \mathit{recruit2}, (k, j)) >)$
3	$\rightarrow \mathcal{I}_F < F, \mathsf{ask}(b, X, \mathit{recruit2}, (j(A, i), k)) >$
1′	after A performs a recruit speech act using the *recruit1* protocol with conversation identifier $(i, \_)$,
2′	if an agent b has done an advertise using *recruit2* with identifier (k, j)
3′	then form the intention to ask B about X with identifier $j(A, i)$, meaning reply to A quoting identifier i
4	$[F, \mathsf{ask}(B, X, \mathsf{recruit2}, (j(A, i), k))]$
5	$\mathcal{I}_B < B, \mathsf{tell}(A, X, \mathsf{recruit3}, (l, i)) >$
4′	after F performs an ask using *recruit2* with identifier $(j(A, i), k)$,
5′	form the intention to tell A about X using *recruit3* with identifier (l, i)
6	$[A, \mathsf{broker}(F, \mathsf{ask}(X), \mathit{broker1}, (i, \_))]$
7	$\exists b, j, k.\mathrm{DONE}(< b, \mathsf{ad}.(F, \mathsf{ask}(X), \mathit{broker2}, (k, j)) >)$
8	$\rightarrow \mathcal{I}_F < F, \mathsf{ask}(b, X, \mathit{broker2}, (j(i), k)) >$
6′	after A performs a broker using *broker1* with identifier $(i, \_)$,
7′	if an agent b has done an advertise using *broker2* with identifier (k, j)
8′	then form the intention to ask b about X with identifier $(j(i), k)$
9	$[B, \mathsf{tell}(F, X, \mathit{broker2}, (k, j(i)))]$
10	$\exists a, l.\mathrm{DONE}(< a, \mathsf{broker}(F, \mathit{ask}(X), \mathit{broker1}, (i, l)) >)$
11	$\mathcal{I}_F < F, \mathsf{tell}(a, X, \mathit{broker1}, (l, i)) >$
9′	after B replies with a tell using *broker2* with identifier $(k, j(i))$,
10′	then there is an agent a who did a broker using *broker1* with identifier (i, l)
11′	and form the intention to tell agent a about X using *broker1* with identifier (l, i)
12	$[A, \mathsf{subscribe}(F, \mathsf{ask}(X), \mathit{subscribe}, (i, j))]$
13	$\mathcal{B}_F \exists b.\mathit{DONE}(< b, \mathsf{tell}(F, X, \_, \_) >$ $\rightarrow \mathcal{I}_F < F, \mathsf{tell}(A, X, \mathit{subscribe}, (j, i)) >$
12′	after A does a subscribe using the *subscribe* protocol with identifier (i, j),
13′	then add the belief that, after any agent tells the facilitator F about the value of X (irrespective of protocol or conversation identifier), then F forms the intention to tell A using the *subscribe* protocol with the identifier (j, i)
14	$[A, \mathsf{recommend}(F, \mathsf{ask}(X), \mathit{recommend}, (i, j))]$
15	$\exists b.\mathrm{DONE}(< b, \mathsf{advertise}(F, \mathsf{ask}(X), \mathit{recommend}, \_) >)$
16	$\rightarrow \mathcal{I}_F < F, \mathsf{reply}(A, b, \mathit{recommend}, (j, i)) >$
14′	after A does a recommend using *recommend* with identifier (i, j),
15′	then if there is an agent b that has advertised $\mathsf{ask}(X)$
16′	then form the intention to tell A about agent b in the *recommend* protocol using the identifier (j, i)

Table 1: Brokerage Intentional Specifications

had performed an advertise). Therefore, we might specify the following:

$[A, \mathsf{broker}(F, \mathsf{ask}(X), \mathit{broker1}, (i, \_))]$
$(\exists b, j, k.\mathrm{DONE}(< b, \mathsf{ad}.(F, \mathsf{ask}(X), \mathit{broker2}, (k, j)) >)$
$\rightarrow \mathcal{I}_F < F, \mathsf{ask}(b, X, \mathit{broker2}, (j(i), k)) >)$
$\vee$
$(\neg\exists b.\mathrm{DONE}(< b, \mathsf{ad}.(F, \mathsf{ask}(X), \mathit{broker2}, \_) >)$
$\rightarrow \mathcal{I}_F < F, \mathsf{tell}(A, \mathsf{not\_advertised}, \mathit{broker1}, (j, i)) >)$

Recall that this is a receiver's side specification interpreted locally (not a system-wide (global) speciifcation), so this formula states that after the broker speech act, if the facilitator knows of no agent that has advertised the service, then it should tell the sender that this is the case. Of course, we also need to update the *broker1* state diagrams to reflect this improvement to the specification.

In addition, from inspection of Table 1, it might be considered odd that agent B needs to advertise both in the *recruit2*, *broker2* and *recommend* protocols. However, these are different protocols, with different state diagrams; furthermore, advertising with the protocol name offers an implicit contract that the agent will understand the different conversation identifiers that will be used in each case. In addition, an agent may want to preserve anonymity, and while be willing to participate in brokerage, it may now want to participate in recruitment.

By specifying the decision-making and error recovery, we also see that the protocols remain incomplete. What action is taken, for example, for an agent B that wishes to withdraw an advertise, or an agent A that wishes to cancel a subscription? To handle this, we might introduce new performatives (withdraw and cancel, say), update the protocols, and introduce extra conditions into the intentional specifications, for example:

$[A, \mathsf{recruit}(F, \mathsf{ask}(X), \mathit{recruit1}, (i, \_))]$
$\exists b, j, k.(\mathrm{DONE}(< b, \mathsf{ad}.(F, \mathsf{ask}(X), \mathit{recruit2}, (k, j)) >)$
$\wedge\neg\mathrm{DONE}(< b, \mathsf{withdraw}(F, \mathsf{ad}.(\ldots), \mathit{recruit2}, (k, j)) >)$
$\rightarrow \mathcal{I}_F < F, \mathsf{ask}(b, X, \mathit{recruit2}, (j(A, i), k)) >$

This is not the first time such problems have been identified with the KQML brokerage protocols [7]. Indeed we are endorsing the work of [7], in that we argue for a formal semantics, a development method, and (ideally) automated tools, in order to support rigorous design allowing development of solutions to such problems.

4. AUCTION PROTOCOLS

In this section, we consider auctions, another type of multi-party agent conversation. We concentrate in particular on what are typically called English Auctions, in which an auctioneer announces a proposed sale price (the asking price) to a group of potential bidders. If one of them accepts the price, the auctioneer increases it by a set amount, and announces a new asking price. If no-one accepts, the goods on auction are sold to the last bidder.

Auction protocols can be distinguished from the brokerage protocols studies in the previous section, in that instead of several one-to-one conversations which are conducted, sequentially or nested, according to different protocols, auction protocols comprise, in effect, several one-to-one conversations being conducted concurrently according to the same protocol.

The techniques used to specify the semantics of an English Auction are a richer representation of state, and parameterised conversation identifiers, as in the previous section. Conversation identifiers in auctions are vector timestamps [11].

4.1 English Auction: State Diagram

The finite state diagram for an English Auction is illustrated in Figure 5. We assume a previous stage of registration to participate and declaration of lots, etc., covered by some alternate protocols. We consider here only the process of a single auction.

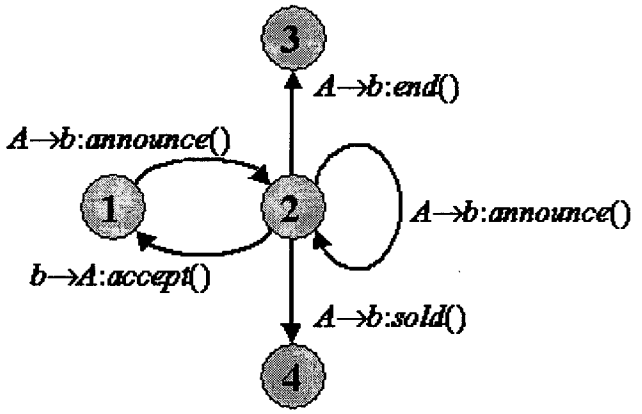

Figure 5: English Auction Protocol

The English Auction Protocol is used between an Auctioneer agent A and a set of buyer agents B. In effect, this translates to a single conversation between A and each agent $b \in B$, following the protocol shown in Figure 5. After the first announce, which is multi-cast to all agents, if no agent bids (accepts), the auctioneer can end the auction. Otherwise, if an agent decides to try to buy the lot at this price, it sends an accept to the auctioneer. The auctioneer accordingly increases the price, and announces this to all agents. This process repeats, until no agent makes a bid (accept). Then the auction is won by the agent that made the last bid. This agent is told of its win by a sold message and all other agents are informed that the auction is at an end.

4.2 Vector Timestamps

The problem with the protocol described in the previous sub- section is that a transition from state 2 is context-sensitive, i.e. for an agent making a bid, it goes into state 1 first; for all other agents, all they 'see' is the new announce. However, these other agents may also have sent accept messages – except accepting the old price and have therefore arrived too late. Therefore, it is necessary to resolve:

1 For the auctioneer, if an accept is in response to the most recent multi-cast announce;

2 For the bidding agents, whether or not it is their bid which has been accepted.

Our solution to this is firstly to use a unique conversation identifier between the auctioneer and each agent, in the style of the previous section. However, in the English Auction Protocol, the conversation identifier is parameterised by the sending agent with a vector timestamp [11] which will enable the agents to determine potential causality.

The idea of potential causality in distributed systems without global clocks is that the ordering of event can only be determined locally, i.e. with respect to any single observer.

If an event e precedes an event f, from one observer's point of view, then e potentially caused f. To determine if one event preceded another, each agent uses its own *vector clock*. Following [11], a vector clock is defined over a group of agents with cardinality n as an (n)-ary vector of natural numbers $v =< a_1, a_2, \ldots, a_n >$. The starting clock of every agent is $< 0, 0, \ldots, 0 >$. Each agent increments its entry in the vector when it performs a local event, i.e. sending or receiving a message. It attaches its entire vector clock as a timestamp to every message it sends out. When it receives a message, it takes the element-wise max of its vector clock and the timestamp on the incoming message, increments its entry in the vector by 1 (for the local event), and sets this value to be its new vector clock.

One vector is defined as later than another if the value of at least one entry in the first vector is greater than the corresponding entry in the second, and no value in the second is greater than the first, i.e. v is later than u if and only if firstly $\forall i.1 \leq i \leq n : u_i \leq v_i$ and secondly $\exists i.1 \leq i \leq n : u_i < v_i$. The notation $u \prec v$ indicates that v is later than u.

Figure 6 shows a message sequence chart for a possible exchange of a number of announce and accept message between an auctioneer A and two bidding agents $b1$ and $b2$ during an English Auction. (The labelling 'mx' on some messages is incidental and used to explain cause and effect between messages below.) Note that after the first announce, both $b1$ and $b2$ accept, but $b1$'s bid arrives first. Therefore this is the accepted bid which causes the second (multi-cast) announcement. Note also that we stipulate that the Auctioneer must 'block' (not read) any incoming messages after an accept, until it has finished the multi-cast announce, i.e. a separate announce message has been sent to each participating agent in B, and so has been assigned a timestamp.

The agents can now use the vector timestamps to determine which announce (potentially) caused which accept, and vice versa, using the two following rules:

- R1 For the auctioneer, if the timestamp on an incoming accept message is later than the auctioneer's vector clock after the last accepted bid (the last accept to cause an announce), then it was caused by the most recent multi-cast announce;
- R2 For a bidding agent, if the timestamp on an incoming announce is later than their vector clock, then the announce was caused by its accept.

Applying these rules to the message sequence chart of Figure 6, the causality relations of messages m1-m7 are summarised below. Rule R1 is applied by the auctioneer to messages m1-m3, and Rule R2 is applied by bidding agents to messages m4-m7.

m1	$120 \prec 000$	accept is potentially caused by most recent (multi-cast) announce
m2	$202 \not\prec 320$	accept not caused by most recent (multi-cast) announce
m3	$524 \prec 320$	accept is potentially caused by most recent (broadcast) announce
m4	$420 \prec 120$	(multi-cast) announce potentially caused by b1's accept (m1)
m5	$520 \not\prec 202$	(multi-cast) announce not caused by b2's accept (m2)
m6	$824 \not\prec 430$	(multi-cast) announce not caused by b1's accept
m7	$924 \prec 524$	(multi-cast) announce potentially caused by b2's accept (m3)

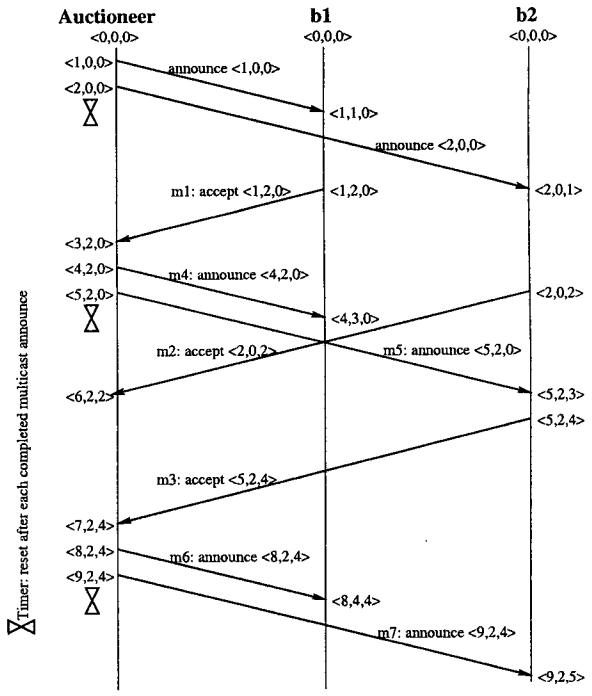

Figure 6: Message Sequence Chart for English Auction

4.3 Intentional Specifications

Finally in this section, we specify the reference implementation model for the English Auction. The problem now is that the change of state is more complex than in ordinary one-to-one protocols, and in the brokerage protocols of KQML. Therefore the simple state representation as an integer is no longer sufficient. The state of an auction is conditional on the announced price, the current winner (performer of the last accept), the set of bidding agents, and the auctioneer's vector clock of the last accepted bid, used in R1.

In fact, there are four parts to the specification that need to be formalised:

1. The change of state of both speaker and hearer after a speech act (message is sent/received);
2. The intentions of hearer in response to a speech act;
3. The (real) time aspects, i.e. if no accept is received within a certain time, then the auctioneer terminates the auction;
4. Rules R1 and R2 upon which 1–3 above are conditional.

For the first item, we require a richer representation of the protocol states, so need the following state variables:

- *buyer* the agent currently winning the auction (i.e. the agent who made the last accepted bid)
- *price* current asking price of the item for auction
- B set of agents bidding in this auction
- *la* timestamp of last accepted bid
- *time* real time remaining before auction is ended

Access and change to a state variable *svar* is indicated by the following notation:

- $conv_a(i).svar = val$ to test the value for equality
- $conv_a(i).svar \leftarrow newval$ to overwrite the current value

The following identifiers are also required:

vc, ts, la	are all vector clocks, where vc is the vector clock of an agent ts is the timestamp on a message la is the vector clock of last accepted bid
$(i(ts), j)$	i is the unique number identifying this conversation for the sender, ts is its vector clock at the time of sending, and j identifies the conversation for the receiver
$conv_a(i)$	returns the protocol state of the conversation identified by i for agent a (as in section 3.1)
$cid_a(i,b)$	returns the conversation identifier used by agent b in the conversation identified (for agent a) by i (as in section 3.2)

Finally, we need a notation to multi-cast a speech act:

$$< s, multicast(\mathsf{perf}, R, C, P, i) >= \\ ;_{r \in R} < s, \mathsf{perf}, C, P, (i(vc), cid_s(i,r)) >$$

This describes the multi-cast of a performative perf, to a set of receivers R, with content C, in protocol P, performed by an agent s as the composition of a sequence of individuals acts to each $r \in R$. Since each agent can only do one act at a time, the specific conversation identifier for each speech act in the multi-cast is constructed from vc is the vector clock at the time of occurrence of the local event (and is incremented after performing each individual announce speech act), and so is different for each $r \in R$.

The English Auction multi-party protocol is formally specified (and paraphrased) by the intentional specifications contained in Table 2 for the communicative acts involved. Note that there are different axioms for both sender (auctioneer) and receiver (bidding agents) for the announce act. Note that we are assuming that vector clocks will be correctly updated to account for the occurrence of local events.

5. SUMMARY AND CONCLUSIONS

In this paper, we have taken a general semantic framework for specifying the semantics of an Agent Communication Language, which concentrated on one-to-one conversations, and extended it to cover multi-party conversations as found in brokerage and auction protocols. The specific extensions introduced were:

- Increased range of the possible intended replies, so that the 'meaning' of a speech act between a speaker and hearer in one protocol could, at the action level, be the intention to perform another speech act in a new protocol with a third party;
- Conversation identifiers that were structures (2-tuples), one element of which was supplied by each party to the speech act, and could be parameterised with additional information. The identifiers were then brought into the semantics at the intentional level rather than being or considered as pragmatics as in the FIPA ACL semantics;
- A richer representation of protocol states, so that states were no longer single integers but could include a number of variables. These variables could change value after either the speaker sent the message, and reflect the fact that both the speaker and hearer are changing state when a speech act is performed.

	Auctioneer's trigger for multi-cast announce
17	$[A, multicast(\mathsf{announce}, B, price, EA, (i, \_))]$
18	$conv(i).time \leftarrow 50$
17′	after A does a multi-cast announce using the English Auction protocol (EA) with identifier $(i, \_)$
18′	reset the *time* state variable to some set number of time units (e.g. 50 in this case)
	Bidding agent's reaction on receiving its (individual) announce
19	$[A, \mathsf{announce}(b, price, EA, (i(ts), j))]$
20	$(vc =< 0, 0, \ldots, 0 >) \vee (ts \prec vc)$
21	$\rightarrow compute\_accept(price)$
22	$\rightarrow \mathcal{I}_b < b, \mathsf{accept}(A, \_, EA, (j(vc'), i)) >$
19′	after A does an announce in the EA protocol with conversation identifier containing timestamp ts
20′	if the vector clock is all zeros (the start of the auction) or rule R2 does not apply (otherwise b is winning)
21′	then if the agent decides to bid at this price ($compute\_accept(price)$ evaluates to true)
22′	then form the intention to send an accept to the auctioneer, vc' being the vector clock of the send local event
	Auctioneer's reaction after receiving an accept
23	$[b, \mathsf{accept}(A, \_, EA, (j(ts), i))]$
24	$conv(i).time > 0$
25	$\rightarrow ts \prec conv(i).la$
26	$\rightarrow conv(i).buyer \leftarrow b$ $\wedge\, conv(i).price \leftarrow conv(i).price + increment$ $\wedge\, conv(i).la \leftarrow vc$
27	$\wedge\, \mathcal{I}_A < A, multicast(\mathsf{announce}, conv(i).B, conv(i).price, EA, i) >$
23′	after b does an accept in the EA protocol with conversation identifier $(j(ts), i)$
24′	if the auction hasn't timed out and been terminated
25′	then if rule R1 applies
26′	update all the relevant state variables
27′	and form the intention to multi-cast the updated price to all the bidding agents
	Auctioneer's triggers for dealing with timeouts
28	$\exists i.conv(i).time < 0 \rightarrow \mathcal{I}_A < A, timeout(i) >$
28′	if there is any conversation identifier i for which the value of the *time* state variable drops below zero, then form the intention to perform a *timeout* action
29	$[A, timeout(i)]$
30	$conv(i).B \leftarrow conv(i).B - \{conv(i).buyer\}$
31	$\wedge\, \mathcal{I}_A < A, \mathsf{sold}(buyer, \_, EA, (i(vc), cid(i, buyer)) >$
32	$\wedge\, \mathcal{I}_A < A, multicast(\mathsf{end}, conv(i).B, \_, EA, i) >$
29′	after a timeout in a particular auction
30′	remove the winner (*buyer*) from set of bidding agents
31′	confirm the sale to the buyer (strictly we should write $conv(i).buyer$ but this is obvious from context)
32′	end the auction with a multicast to the rest

Table 2: Auction Intentional Specifications

We then used these extensions to give precise, detailed formal specifications (reference implementation models) of the meaning of particular speech acts in the context of a specific protocol. The intention is then, of course, for developers to be able to implement such specifications in agents which could then inter-operate in any open multi-agent system that was using this communication language (performative, protocols and specifications) for uni-cast and multi-cast messages. Broadcast messages are currently out of the scope

of this framework.

We believe that analysing multi-party conversations in terms of their individual components makes the requirements on each interaction clearer, and so assists in determining where the dependencies occur (e.g. sequence, cross-reference, and so on). Having done this analytic design, it is of course feasible to re-constitute the original presentation, if it helps the designer. It is also our contention that the performative sematics is clearer from being defined in context of use (pace Wittgenstein) rather than being defined, syntactically or semantically, as a composite or complex speech act [1; 4]. We would further argue that if conversation identifiers are going to be used in an ACL, then it is worth using them, mechanistically as here, for guaranteeing unique conversations and for passing additional information, to be interpreted in context. However, in [1] it was shown how the 'commitments' in composite acts could be derived from the conditions on the primitive ones, and one area for further research is how such commitments are preserved across linked protocols.

We also observed that the auction and brokerage protocols have been used widely in MAS, although formal specifications appeared to be in short supply. The FIPA experience, though, and the work in this paper, suggest that achieving unambiguous specifications for multi-party protocols requires a wide range of considerations to be met. Indeed, the logical specification language required to capture all the different aspects (time, state, events, intentions, axioms, etc.) needs to be extremely powerful and expressive, and correspondingly complex. It is therefore clear that when it comes to providing standard specifications, we are treading a fine line between being too dense and arcane, and being useful and understandable. It remains the case, too, that the formal specification language here has an intuitive but somewhat loose semantics, and this is being addressed in current research.

We maintain though that a systematic approach to design at different levels of 'meaning' offers definite development advantages, in terms of clearer interfaces, open standards, modularity and re-use. Furthermore, the design process can expose new problems. For example, the Facilitator tell in lines 12-13 of Table 1 have F telling X, without necessarily believing X itself. In open systems it might be that F is liable for what it 'says', and that therefore quoting is required, or some other safeguard.

The way the English Auction protocol has been specified and implemented within our framework may point the way forward. We have had to make use of variables which control how the agents follow the protocol. There is also scope for identifying the various roles that the participants play at certain points of the protocol according to the state. It therefore appears that there is an underlying richer state description which cannot be represented in the finite state diagram. Ideally we would like to have an explicit representation of roles and control variables within the general framework, where the action level diagrams and semantics use these roles, rather than specifying them solely in the logical implementation stage. This will require another extension of the semantic framework and is the subject of ongoing research, but the result will be a clearer separation of the constituent 'meanings' of speech acts and protocols, plus their dependencies and relations, which combined together provide a more accessible specification.

6. ACKNOWLEDGEMENTS

This work has been undertaken in the context of the UK-EPSRC/Nortel Networks funded Project CASBAh (Common Agent Service Brokering Architecture, GR/L34440), and the EU-funded MARINER Project (ACTS 333), and the support from these funding bodies is gratefully acknowledged. We are also very appreciative of the detailed and constructive comments of the anonymous reviewers.

7. REFERENCES

[1] P. Cohen and H. Levesque. Communicative actions for artificial agents. In V. Lesser, editor, *Proceedings ICMAS95*. AAAI Press, 1995.

[2] R. Cost, Y. Chen, T. Finin, Y. Labrou, and Y. Peng. Using Colored Petri Nets for conversation modeliing. In F. Dignum and B. Chaib-draa, editors, *IJCAI'99 Workshop on Agent Communication Languages*. Stockholm, Sweden, 1999.

[3] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. Bradshaw, editor, *Software Agents*. MIT Press, 1995.

[4] FIPA. FIPA'97 specification part 2: Agent communication language. FIPA (Foundation for Intelligent Physical Agents), http://drogo.cselt.stet.it/fipa/, 1997.

[5] J. Pitt and A. Mamdani. Designing agent communication languages for multi-agent systems. In F. Garijo and M. Boman, editors, *Multi-Agent System Engineering MAAMAW'99*, volume LNAI1647, pages 102–114. Springer-Verlag, 1999.

[6] J. Pitt and A. Mamdani. A protocol-based semantics for an agent communication language. In *Proceedings 16th IJCAI'99*, pages 485–491. Morgan-Kaufmann, 1999.

[7] I. Smith, P. Cohen, J. Bradshaw, M. Greaves, and H. Holmback. Designing conversation policies using joint intention theory. In Y. Demazeau, editor, *Proceedings ICMAS98*. IEEE Press, 1998.

[8] Y. Takada, H. Iciki, M. Okada, and T. Mohri. Agent brokerage proposal to cfp6. Fujitsu Laboratories Ltd., Fukuoka, Japan. FIPA CFP6_003, 1999.

[9] Y. Takada, H. Iciki, M. Okada, and T. Mohri. A proposal on agent brokerage. Fujitsu Laboratories Ltd., Fukuoka, Japan. FIPA CFP5_008, http://www.fipa.org/Common_docs/cfp5_008.html, 1999.

[10] FACTS. FIPA Advanced Communication Technologies and Services. EU ACTS project AC317. http://www.labs.bt.com/profsoc/facts, 1999.

[11] M. Venkatraman and M. Singh. Verifying compliance with commitment protocols: Enabling open web-based multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 2(3):pp217–236, 1999.