An Operational Semantics for Intentional Specifications of Agent Behaviour in Communication Protocols

Jeremy Pitt
Intelligent & Interactive Systems, Department of Electrical & Electronic Enginering
Imperial College of Science, Technology & Medicine, Exhibition Road, London, SW7 2BT, UK
j.pitt@ic.ac.uk

ABSTRACT
Protocols have been advocated as the basis for the semantics of a standard Agent Communication Language. For any representation of the protocols, intentional specifications are required to formalise an agent's "thought processes" for doing and responding to speech acts in the context of these protocols. For this purpose, we have been using a style of modal action logic, but with a rather loose semantics, so that although the intended meaning was intuitively clear, the formal statements still lacked a precise characterisation. The main contribution of this paper is to give a formal semantics to our specification language. This is done not model-theoretically, but operationally, through a transformation into Prolog clauses. These are embedded in the interpreter (reasoning system) of a BDI (Beliefs-Desires-Intentions) Agent. Based on a small Agent Communication Language, we give illustrative 'animations' of speech acts using this architecture: firstly for acts performed in isolation and then in the context of a protocol. We conclude that the formal semantics for the intentional language is one of the last pieces of the jigsaw required to support the sound engineering of agent communications in multi-agent systems.

1. INTRODUCTION
In previous work, we have advocated the use of protocols as the basis for a standard semantics for Agent Communication Languages [18], whether these are specified as finite-state diagrams, as colored Petri Nets [3], or by using Unified Agent Modeling Language (UAML [14]).

The richer the protocol description language, the more complex the interactions that can be described (although requiring in turn a more more complex semantics). Underpinning any such representation formalism for protocols, though, there is a requirement for intentional specifications of an agent's "thought processes" for doing and responding to speech acts (in the context of these protocols) [17]. For this purpose, we have been using a style of modal action logic, but with a rather loose semantics, so that although the intended meaning of the formal statements was, at least, intuitively clear, it still lacked a precise formal characterisation.

The main contribution of this paper is to give a formal semantics to our intentional specification language. This is given not model-theoretically, as per Rao and Georgeff's extension of the logic CTL* [20], but operationally, through a transformation into Prolog clauses. These are embedded in the interpreter (reasoning system) of a BDI (Beliefs-Desires-Intentions) Agent [21]. We give illustrative 'animations' of speech acts transmitted between agents based on this BDI architecture, firstly for a speech act performed in isolation and then for acts performed in the context of a protocol.

Our argument then is that we have: (1) a small ACL which can serve as a standard, with a way of accounting for its semantics in an externally verifiable way; (2) a development method and formal specification language for extending the language for particular applications; and (3) a reference architecture that can serve as an implentation model, that is actually consistent with the given semantics and useful to agent developers. We conclude that the provision of a formal semantics for the intentional language is one of the last pieces of the jigsaw required to support the sound engineering of agent communications in multi-agent systems.

The current paper re-uses, revisits and revises earlier work. It is still based on a general semantic framework for describing the semantics of Agent Communication Languages in terms of protocols which has been described elsewhere [18, 17]. Some familiarity with this work is presumed and the framework will not be repeated here. Certain enhancements (conversation identifiers and state variables) to deal with multi-party protocols are described in [16]. Again, we will not motivate these enhancements here, and will take them as given. This paper also illustrates our general method of describing an ACL through protocols and intentional specifications, the semantics of the latter being the main fosuc of the current work. Therefore we briefly review the method in Section 2. Section 3 focuses on the intentional specification language, its syntax, and its operational semantics as given by a transformation into Prolog. Based on a small ACL called sACL introduced in Section 4, Section 5 gives sample 'animations' of the Prolog specifications as interpreted by a BDI agent. We conclude in section 6 with some comments on engineering multi-agent systems and directions for further research.

2. THE ACL DEVELOPMENT METHOD

We define an ACL by a 3-tuple:

$$< \mathsf{Perf}, \mathsf{Prot}, \mathit{reply} >$$

where Perf is a set of performatives, Prot is a set of protocol names, and *reply* is a function that specifies, for each performative and protocol (state), what are the allowed (i.e. the range of possible) replies. This effectively specifies a set of finite state diagrams with arcs labeled by performatives and states labeled by which agent's turn it is to 'speak' in the conversation.

This approach allows us to define a class of ACLs, rooted on a well-defined core language. We envisage starting from a set of core performatives and protocols (i.e. performatives whose intended meaning is intuitively clear and protocols with some common functions). This would be the 'standard' ACL, the root of this class of ACLs, which could then be extended within the same semantic framework for particular applications. There are three directions in which an ACL core could be extended, as illustrated in Figure 1, with each axis corresponding to one of the three levels of meaning (action, intentional, and content) identified in [18]:

Generalization at the action level. More performatives and new protocols can be defined to create a new ACL where there were general interactions not covered by the standard. For example, auction protocols are a common requirement in some multi-agent systems, and new protocols could be added for the particular type of auction demanded by an application;

Specialization at the intentional level. Additional constraints can be added to transitions, and/or richer description of states can be specified. In particular we can refer to the intentional state (beliefs, desires and intentions) for how these components of agent 'mental affect' *may* be affected by receiving a message, and what might be done in reply;

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. The decision-making that characterises whether to reply with performative X or performative Y from the same state may then be dependent on very different functions.

To give a illustrative example of what we have in mind, Figure 2 specifies a finite-state diagram for the FIPA'97 protocol FIPA-request (see [15] for further explanation).

Agents will have reasons for using this protocol, and we use our intentional specification language for stating why an agent should initiate a conversation (what we call a *trigger*), and for stating specifications of what an agent 'should' do, in response to an incoming message (what we call a *tropism*).

The intentional specification for triggering (initiating) a conversation according to the FIPA_request protocol could then

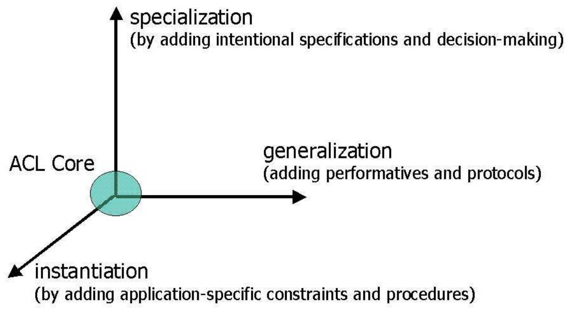

specialization
(by adding intentional specifications and decision-making)
ACL Core
generalization
(adding performatives and protocols)
instantiation
(by adding application-specific constraints and procedures)

Figure 1: Framework for Developing ACLs

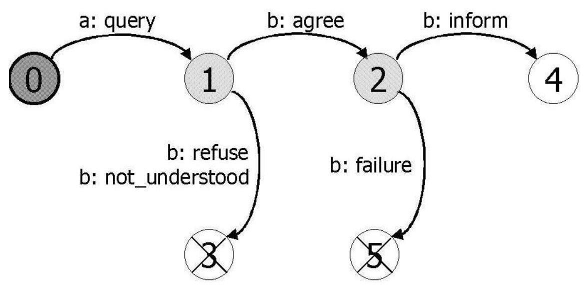


Figure 2: Finite State Diagram for FIPA-request Protocol

be as shown in the first line of Table 1. (Note that this is *not* in the FIPA ACL specification or indeed FIPA's current way of specifying its protocols.) This states that if agent s desires ($\mathcal{D}$) that action A be done, and believes that r is capable of A, then s will form the intention to request r to do A, using the *FIPA_request* protocol. On receiving such a speech act, r's behaviour could then be specified as also shown in Table 1.

This specification states that: after s requests r to do A, then if r is capable of A and it is ok for r to do A, then r intends to tell s that it agrees to do A, intends to do A, and believes that after it does A the action it should take are: if A is done successfully, then inform s that DONE(A) is true, otherwise inform s that the attempt to do A failed. If it was not ok for r to do A or it was not capable of doing A, then r should intend to tell s that r refuses to do A or tell s that r does not understand the request, respectively.

It can be seen that it is easy to paraphrase the intended meaning of these specifications, but for automation, verification, unambiguous interpretation, etc., a formal semantics for this intentional language is required. In particular, we are concerned with treating the specifications as axiom schemas, i.e. always true for whatever values of s, r, A, etc. are actually given.

We note also that the implementation of *Capable*, *ok_to_do*, and success or failure of actions are all application dependent. Therefore we can specify the actual decision making needed to compute the truth- or false-hood of these predicates. This is the process we call instantiation. However, it

$$
\begin{array}{ll}
\text{FIPA\_request trigger} & \mathcal{D}_s \mathrm{DONE}(A) \wedge \mathcal{B}_s\, \mathit{Capable}(r, A) \rightarrow \mathcal{I}_s \ll s, \mathsf{request}(r, A, \mathit{FIPA\_request}) \gg \\
\hline
\text{FIPA\_request tropism} & [s, \mathsf{request}(r, A, \mathit{FIPA\_request})] \\
 & \quad (\mathit{Capable}(r, A) \rightarrow \\
 & \qquad (\mathit{ok\_to\_do}(A) \rightarrow \\
 & \qquad\qquad \mathcal{I}_r \ll r, \mathsf{agree}(s, A, \mathit{FIPA\_request}) \gg \\
 & \qquad\quad \wedge\ \mathcal{I}_r \ll r, A \gg \\
 & \qquad\quad \wedge\ \mathcal{B}_r [r, A](\mathit{success}(A) \rightarrow \mathcal{I}_r \ll r, \mathsf{inform}(s, \mathrm{DONE}(A), \mathit{FIPA\_request}) \gg) \\
 & \qquad\qquad\quad \vee (\mathit{failed}(A) \rightarrow \mathcal{I}_r \ll r, \mathsf{failure}(s, A, \mathit{FIPA\_request}) \gg)\ )\ ) \\
 & \qquad \vee (\neg \mathit{ok\_to\_do}(A) \rightarrow \mathcal{I}_r \ll r, \mathsf{refuse}(s, A, \mathit{FIPA\_request}) \gg) \\
 & \quad \vee (\neg \mathit{Capable}(r, A) \rightarrow \mathcal{I}_r \ll r, \mathsf{not\_understood}(s, A, \mathit{FIPA\_request}) \gg)
\end{array}
$$

Table 1: Intentional Specification for FIPA_request

is important to note that the *normative* part of the standard applies to the ACL specification only, i.e. in the specification of performatives, protocols and appropriate replies. The intentional language and its semantics (as described in the next section) are *informative*, that is, it is an illustrative specification of *how* an agent could be implemented (which implementation complies with the requirements of the normative standard), and not a specification of what must be implemented. Therefore we do not constrain agents to be implemented as BDI agents, we insist on compliance only with regard to external (observable) behaviour, and resolve the tension between the intentional semantics, the standard and the way that an agent is implemented.

3. THE INTENTIONAL SPECIFICATION LANGUAGE

3.1 Triggers and Tropisms

We follow many existing works which use beliefs, desires and intentions (BDI) to characterise the semantics of performatives [22, 2, 11].

However, we use such BDI models to characterise the intentional meaning as only one component of the overall meaning of a speech act. The intentional specifications are used to state what we call triggers and tropisms [17]. The idea is that triggers are the combination of beliefs and desires thar produce intentions (to do actions), and that tropisms are the effects on beliefs and desires that results from executing those intentions.

Notice that triggers and tropisms specified like this are local to an agent, and while they may be symmetric between a sender and receiver agent in a multi-agent system, they may well not be (for example in an open system with agents deployed by heterogeneous organizations). Therefore, we have the following categorization:

	trigger	tropism
sender's side	sender's actual motivation for doing an action	sender's belief about expected outcome of action on receiver's belief state
receiver's side	receiver's belief about motivation (sender's belief state) for doing an action	actual result (receiver's actual change of belief state) of doing action

So while the sender can be certain about why it did an action and a receiver certain about what happened to it as a result, each side infers beliefs about what happened and why it was done. If the local speciifcations are made 'public', then these are good inferences, otherwise they are beliefs which may have to be revised.

The formal syntax of this language is a first-order modal logic with relativised belief, desire, and intention (to) modalities $\mathcal{B}_a$, $\mathcal{D}_a$ and $\mathcal{I}_a$ respectively; action formulas written $\ll a, A \gg$ and a DONE operator on action formulas; and parameterised action modalities $[a, A]$ (read as "after agent a does action A"). We use the notation $\mathcal{K}_a\phi$ for knowledge (one way or another), i.e. $\mathcal{K}_a\phi \equiv \mathcal{B}_a\phi \vee \mathcal{B}_a\neg\phi$.

3.2 Semantics

To give a formal specification of intentional behaviour, we want to write axioms of the form (for any agent a):

$$\models \mathcal{B}_a\phi \wedge \mathcal{D}_a\psi \rightarrow \mathcal{I}_a \ll a, A \gg$$
$$\models [a, A]\chi$$

Here, $\mathcal{B}$, $\mathcal{D}$ and $\mathcal{I}$ are relativised (agent) modalities for beliefs, desires and intentions, and $[a, A]$ is an agent-action modality. The intuitive reading of these axioms is then firstly, that if an agent believes ϕ and desires ψ, then it will form the intention to perform action A. Secondly, that after agent a performs action A, χ holds.

However, the semantics of this language is rather more problematic, especially as there are certain mechanical requirements to take into account. Furthermore, we often need to write axioms of the form:

$$\models [a, A]\phi \rightarrow \chi$$

with the intended reading: that after agent a does action A, whichever agent "sees" the action (i.e. receives the message if it was a speech act) *will try to make* $\phi \rightarrow \chi$ true. In such cases, ϕ is some kind of pre-condition and χ is some formula involving $\mathcal{B}$, $\mathcal{D}$ and $\mathcal{I}$ modalities, which, if the precondition holds, the agent should try to make true. This could be modelled by a deontic modality, for example, the agent could be obliged to make $\phi \rightarrow \chi$ true, but there are several ways of doing this (e.g. by making the precondition false, which is not what we want). However, it is really the interpretation of the $\rightarrow$ connective that needs to be changed, as it is not classical 'material implication' but more an interpretation of 'causality' that is required.

We will not therefore develop a model-theoretic semantics for this logic here. Instead, we give this language a procedural semantics via a mapping into Prolog (cf. [5, 6]). (Note that this approach has much in common with the KQML semantics proposed in [10], which was based on Definite Clause Grammars (DCG)).

The basic mapping of triggers and tropisms into Prolog is as follows. A trigger of the form:

$$\models \mathcal{B}_a\phi \wedge \mathcal{D}_a\psi \rightarrow \mathcal{I}_a \ll a, A \gg$$

maps onto Prolog of the form:

```
trigger( A ) :-
    believes( a, $\phi$ ),
    desires( a, $\psi$ ),
    assert( intends( a, A ) ).
```

and a tropism of the form:

$$\models [Agent, Action]\phi \rightarrow \chi$$

as seen by an agent b maps onto Prolog of the form:

```
process_message( Agent, Action ) :-
    believes( b, $\phi$ ),
    assert( $\chi$ ).
```

These would be used in an algorithmic specification of a BDI agent with a control cycle (cf. [21]):

```
bdi_control_loop :-
    read_one,
    trigger_all,
    execute_one,
    bdi_control_loop.
```

where `read_one` reads one incoming message off a buffered input queue, `trigger_all` finds all of those triggers whose pre-conditions are satisfied, and `execute_one` selects one intention and executes it.

Further specifications for `trigger_all` and `read_one` are as follows (note we reserve the right for the reasoning system to filter intentions so that not all triggered intentions will be executed, e.g. if the same goal triggered multiple intentions):

```
trigger_all :-
    findall(    I,
                trigger( I ),
                Triggered ),
    filter( Triggered ).

read_one :-
   read_message_queue( (S,SpAct) ),
   SpAct =..  [Perf,R,C,L,O,Prot,(I,J),T],
   generate_convid( J ),
   conv_r( J, State ),
   change_state( State, Perf, Prot, NewS ),
   overwrite_convr( J, NewS ),
   assert( 'DONE'( S, SpAct ) ),
   process_message( (S, SpAct) ).
```

The operational reading of `read_one`, for a receiving agent r, is as follows. First, a message is read off the incoming message buffer: this is taken to be a Prolog term akin to the form of the intention (i.e. $\mathcal{I}_a \ll a, A \gg$ for some agent a, a = `Sender`, A = `SpeechAct`) that was executed. The speech act is then decomposed into its constituent parts, and a conversation identifier is generated: if this is a new conversation `J` is a variable, so it is assigned the value of the conversation count, which is then incremented by one (for full details see [16]).

The state of the conversation is then determined using the relativised function $conv_r$ (cf. [16]). Again, if this was a new conversation initiated by the sender then the start state will be 0 (note we only need the receiver's component of the conversation identifier to determine the state, since this is locally unique, the conversation identifier as a whole is unique between agents). The new state of the protocol is then updated according to the finite state diagram, overwriting the $conv_r$ function for this parameter. The agent then asserts (that it believes) DONE($\ll a, A \gg$), and then it processes the message (using the mapping of the tropism 'axioms', as described above.

4. A SMALL ACL

In this section, we describe a small ACL called sACl. We will define this ACL as proposed in [18], as a 3-tuple. The first component of the tuple names the performatives in the langauge, the second component names the protocols, and the third component is the *reply* function. This takes as input a protocol, protocol state and a performative, ad returns as result a set of performative/protocol pairs which can be used by an agent in reply to the original performative.

4.1 Core-sACL

Let core-sACL be the 3-tuple $<\mathsf{Perf}, \mathit{Prot}, \mathit{reply}>$ where:

$$\begin{aligned} \mathsf{Perf} &= \{\mathsf{inform}, \mathsf{query}, \mathsf{command}\} \\ \mathit{Prot} &= \{\textbf{no\_protocol}\} \\ \mathit{reply}(\mathsf{inform}, \textbf{no\_protocol}, \perp) &= \{\mathsf{null}\} \\ \mathit{reply}(\mathsf{query}, \textbf{no\_protocol}, \perp) &= \{\mathsf{inform}\} \\ \mathit{reply}(\mathsf{command}, \textbf{no\_protocol}, \perp) &= \{\mathsf{inform}, \mathsf{null}\} \end{aligned}$$

This defines what we call the action level semantics of core-sACL. The *reply* function deines the space of possible replies to speech acts (especially in the context of protocols), which can be used to identify finite state diagrams representing those protocols.

Table 2 shows a possible intentional semantics using our specification language. The beliefs and desires that trigger the intention to perform each of the three specified speech acts, for a given sending agent s and an intended receiving agent r.

These specifications then determine which performative to use in a reply in order to comply with the action level semantic definition, the extra information that is required to parameterise the performative to turn it into a speech act, and how the information state of the receiving agent changes after the speech act.

Note, however, that this is a 'non-binding' specification only, and other agents may respond differently provided they comply with the action level semantic definition. This is why we

Trig1	$\mathcal{B}_s p \wedge \mathcal{D}_s \mathcal{B}_r p \rightarrow \mathcal{I}_s \ll s, \mathsf{inform}(r,p) \gg$
Trig2	$\mathcal{D}_s \mathcal{K}_s p \wedge \mathcal{B}_s \mathcal{K}_r p \rightarrow \mathcal{I}_s \ll s, \mathsf{query}(r,p) \gg$
Trig3	$\mathcal{D}_s \mathrm{DONE}(A) \wedge \mathcal{B}_s\, cando(r,A) \rightarrow \mathcal{I}_s \ll s, \mathsf{command}(r,A) \gg$
Trop1	$[s, \mathsf{inform}(r,p)]\mathcal{B}_r p$
Trop2	$[s, \mathsf{query}(r,p)](\mathcal{B}_r p \rightarrow \mathcal{I}_r \ll r, \mathsf{inform}(s,p) \gg) \vee (\mathcal{B}_r \neg p \rightarrow \mathcal{I}_r \ll r, \mathsf{inform}(s, \neg p) \gg)$
Trop3	$[s, \mathsf{command}(r,A)]\mathcal{I}_r \ll r, A \gg \wedge [r,A](\mathrm{DONE}(A) \rightarrow \mathcal{I}_r \ll r, \mathsf{inform}(s, \mathrm{DONE}(A)) \gg)$

Table 2: sACL Intentional Semantics

describe the 3-tuple as specifying *the* action level semantics, as all agents 'talking' core-sACL should comply with this specification. The specification in Table 2 is *an* intentional semantics as different agents may act and re-act in different ways (i.e. it will depend on how the agent has actually been implemented).

4.2 Specialization

Communication often takes place in the context of a conversation and/or a social situation. This section we consider even isolated ('one-off') interactions peech acts in the context of a dynamic social context (cf. [1, 23]), to demonstrate the idea of specialization. To do this, we modify the triggers and tropisms for the command speech act, and introduce a new performative request (this is actually what we call generalization, as discussed in the next section.

One of the features of being an agent is that the process is autonomous. This implies that agents make their own decisions on whether to co-operate, and of course, whether or not they take actions that they were asked or told to do. Therefore, in a multi-agent system, in order to co-operate effectively, there may be some requirement to negotiate authority relationships for the suspension of autonomy. So one agent s might establish an empowerment to command r to do an action A, otherwise it would have to request that r to do A (which might be refused). For this, we would to generalise core-sACL with the new request performative, and modify the *reply* function as appropriate. Then the two alternative approaches to delegation (telling to and asking to) could be represented by:

$$\mathcal{D}_s \mathrm{DONE}(A) \wedge \mathcal{B}_s\, cando(r,A) \wedge \mathcal{B}_s\, authority(s,r) \rightarrow \\ \mathcal{I}_s \ll s, \mathsf{command}(r,A) \gg$$

and:

$$\mathcal{D}_s \mathrm{DONE}(A) \wedge \mathcal{B}_s\, cando(r,A) \wedge \neg \mathcal{B}_s\, authority(s,r) \rightarrow \\ \mathcal{I}_s \ll s, \mathsf{request}(r,A) \gg$$

For receiving agent r, the action modality for a command might be specified as:

$$\begin{array}{l} [s, \mathsf{command}(r,A)] \\ \quad (\mathcal{B}_r\, authority(s,r) \rightarrow \\ \quad\quad (\mathcal{I}_r \ll r, A \gg \wedge [r,A](\mathrm{DONE}(A) \rightarrow \\ \quad\quad\quad \mathcal{I}_r \ll r, \mathsf{inform}(s, \mathrm{DONE}(A)) \gg))) \\ \quad \vee\ (\mathcal{B}_r \neg authority(s,r) \rightarrow \\ \quad\quad \mathcal{I}_r \ll r, \mathsf{refuse}(s,A) \gg) \end{array}$$

and for a request as:

$$\begin{array}{l} [s, \mathsf{request}(r,A)] \\ \quad (\mathcal{D}_r \mathrm{DONE}(A) \rightarrow \\ \quad\quad (\mathcal{I}_r \ll r, A \gg \wedge [r,A](\mathrm{DONE}(A) \rightarrow \\ \quad\quad\quad \mathcal{I}_r \ll r, \mathsf{inform}(s, \mathrm{DONE}(A)) \gg))) \\ \quad \vee(\neg(\mathcal{D}_r \mathrm{DONE}(A) \rightarrow \\ \quad\quad \mathcal{I}_r \ll r, \mathsf{refuse}(s,A) \gg)) \end{array}$$

In the former case r will only do the action A if the appropriate authority exists, in the latter case r will only do A if it fits in with r's own desires.

The summary of all this discussion is that the beliefs, desires and intentions of the sending receiving agents are external to the semantics of sACL, because the required combinations differ according to the agents, applications and domains in which the multi-agent system is set. There are also 'social' factors which have to be considered. We conclude that it is impossible to produce a single ACL whose semantics is defined in terms of the intentional stance. However, we can define a semantics for an Agent Communication Language where the normative part is based on a commitment to reply and an informative specification is provided for a particular application. In the next section, we consider the affect of adding protocols to sACL.

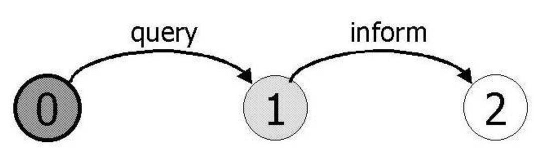
4.3 Generalization

In this section, we generalise core-sACL to include four simple protocols, which also entails introducing a number of new performatives. Let sACL be the 3-tuple $<\mathsf{Perf}, \mathsf{Prot}, reply>$ given by specifying:

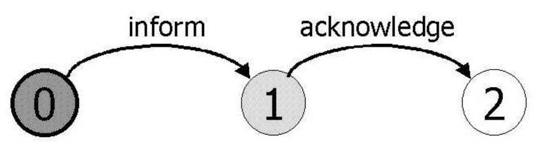
$$\begin{array}{rcl} \mathsf{Perf} & = & \{\mathsf{inform}, \mathsf{query}, \mathsf{command}, \mathsf{acknowledge}, \\ & & \mathsf{end}, \mathsf{request}, \mathsf{agree}, \mathsf{cancel}, \mathsf{fail}, \mathsf{null}\} \\ \mathsf{Prot} & = & \{yesno\_query, datasync, commitment, \\ & & continuous\_update, \mathbf{no\_prot}\} \end{array}$$

The finite state diagrams for the four protocols are illustrated in Figure 3. The corresponding extension to the *reply* function can be constructed from the finite state diagrams.

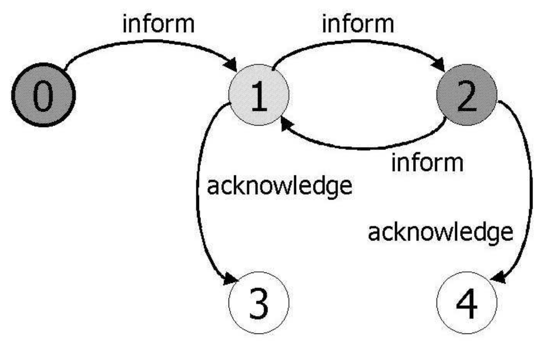
The notation used in Figure 3 is as follows. States are numbered, and arcs are labelled with the speech act that causes the transition between states. The state with the heavy border (state 0) is the start state. States labelled with a tick or cross are end states, with the conversation terminating successfully or not respectively. The different shading on each state indicates which agent's turn it is to continue the dialogue, so here we are specifying turn-taking conversations between only two agents. The notation needs to be



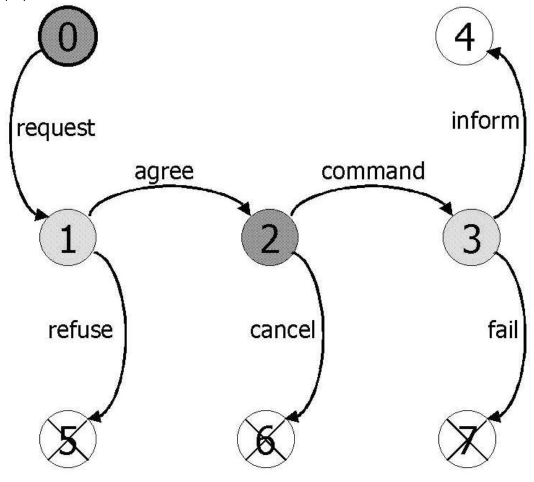
(a): yesno_query protocol



(b): confirmation protocol



(c): data synchronization protocol



(d): commitment protocol

Figure 3: Simple protocols for sACL

extended to accommodate conversations involving multiple participants and/or sequences from one agent. There are other extensions that could also be envisaged: for example, making 'terminate' and 'interrupt' available from any state. These would be a sort of meta-performative which respectively end or change the state of the conversation.

These protocols can be used by one agent to communicate with another in order to:

- query another agent about the value of some proposition (note this is protocol instance of the expected behaviour for one shot query speech acts);
- inform an agent of some proposition and receive an acknowledgement (of belief and/or receipt);
- continuously update the other agent about the value of some proposition;
- ask an agent if it will undertake some action, and, if it is agreeable, to tell (command) it to do that action. The final communication will be the result of doing the action (if it succeeds or fails).

It is now straightforward to construct the actual *reply* function from the finite state diagrams. Furthermore, each protocol can be supplemented by an additional specification (which may be intentional) of a version of the requisite functions which can be implemented by an agent in order to comply with this action level semantics. However, agents are not constrained to implement this specification. Each agent may have its own way of doing things and can do that, provided that its behaviour (in terms of responses to received messages) complies with the action level semantics.

5. REFERENCE ARCHITECTURE

In this section, we discuss an operational model of the BDI agent architecture as suggested in [9], enhanced to accommodate BDI-reasoning about agent–agent communication protocols based on the semantics described in the previous section.

5.1 The BDI Architecture

Kinny *et al.'s* [9] BDI-agent architecture consists of the modules illustrated in Figure 4. Here, the belief database contains facts about 'the world'; the desires module contains goals to be realized; the plan library consists of plans, which are sequences of actions which achieve goals; and the intention structures contains instances of those plans chosen for execution (and currently being executed). The interpreter executes intentions, updates beliefs, modifies goals, and chooses plans.

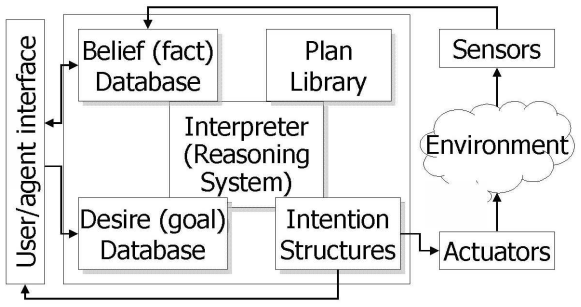


Figure 4: BDI Agent Architecture

The interpreter execution cycle is as follows: At time t:

certain goals are established, and certain beliefs are held. Event(s) occur that alter the beliefs or modify goals, and the new combination of goals and beliefs trigger action plans. One or more action plans are selected and placed on the intention structures. An executable plan is selected and one step is executed, whereby the agent performs the action. Performing the action changes the environment and may establish new goals and beliefs, and so the interpreter execution cycle starts again.

The way we envisaged this architecture working in conjunction with the protocol-based semantics of sACL discussed in the previous two sections is based on the following ideas:

- that the protocols are plans, and the appropriate representation is stored in the Plan Library;
- that a conversation is an instance of a plan, and all the conversations the agent is currently engaged in are stored in the Intention Structures;
- that beliefs and desires are used to trigger plans, i.e. to initiate a conversation by starting an instance of the appropriate protocol;
- that the *select* and *add* functions are used to determine how to progress a conversation according to the plan (i.e. within the remit of the *reply* function).

In the rest of this section, we demonstrate these ideas by animating the intentional specifications of section 5.4 for performatives used on their own and in the context of a protocol, using the execution cycle described in section 4.2.

5.2 Performative Semantics

We can show that the intentional specification of the performative semantics of Table 2 complies with the action-level meaning of sACL as a 3-tuple, by animating a simple question/answer sequence. Suppose that for every agent a, we have the triggers and tropisms schemata implemented as in Table 2.

Then, let agent s and agent r have the following beliefs, desires, and intentions:

$$\begin{array}{llll} \mathcal{B}_s: & \mathcal{K}_r p & \mathcal{B}_r: & \neg p \\ \mathcal{D}_s: & \mathcal{K}_s p & \mathcal{D}_r: & \emptyset \\ \mathcal{I}_s: & \emptyset & \mathcal{I}_r: & \emptyset \end{array}$$

Now (given an execution cycle of from Section 4.2), the combination of s's beliefs and goals in `trigger_all` will satisfy (instantiate) *Trig1*, so s forms the intention:

$$\mathcal{I}_s \ll s, \mathsf{inform}(r, p) \gg$$

Assuming that `execute_one` returns this intention as the selection for execution, when this action (speech act) is executed, the message is sent and received by r, which in its execution cycle will read and process the message using `read_one`. Its reaction is conditioned by *Trop2*, and the extra inference from the disjunction in *Trop2* creates the goal $\mathcal{D}_r\mathcal{B}_s\neg p$. `Trigger_all` for r now generates a new intention from *Trig2*, leaving the situation as follows:

$$\begin{array}{llll} \mathcal{B}_s: & \mathcal{K}_r p & \mathcal{B}_r: & \neg p \\ \mathcal{D}_s: & \mathcal{K}_s p & \mathcal{D}_r: & \mathcal{B}_s \neg p \\ \mathcal{I}_s: & \emptyset & \mathcal{I}_r: & \mathcal{I}_r \ll r, \mathsf{inform}(s, \neg p) \gg \end{array}$$

Assuming that this is r's only intention, it will be selected by `execute_one`. The performance of the inform speech act by r now has two effects. The first is to discharge the the goal because, using *Trop1* from r's point of view, r's expected outcome of its action is $\mathcal{B}_s p$, as required. The second is to discharge the original goal of s, because, again from *Trop1* (but from s's point of view), $\mathcal{B}_s\neg p$ becomes true. This is enough to satisfy $\mathcal{D}_s\mathcal{K}_s p$.

With a closed world assumption, we see that every query will be followed by an intention to perform an inform, and according to the ACL definition:

$$\mathit{reply}(\mathsf{inform}, \textbf{no\_prot}, \bot) = \{(\mathsf{inform}, \textbf{no\_prot})\}$$

The same animation can be applied to the command specifications, whereupon if the commanded action is performed successfully, the intention to reply with an inform is generated.

5.3 Protocol Semantics

Ignoring for now issues like language, time, ontology and protocol, an agent designer could specify an agent r's behaviour for reacting to an inform message with content ϕ from an agent s using the *datasync* protocol to be:

$$\begin{array}{ll} \multicolumn{2}{l}{[s, \mathsf{inform}(r, \phi, \mathit{datasync})]} \\ & \mathcal{B}_r\phi \rightarrow \mathcal{I}_r \ll r, \mathsf{acknowledge}(s, \phi, \mathit{datasync}) \gg \\ \vee & \mathcal{B}_r\phi^c \rightarrow \\ & \quad (\mathit{bel\_revise}(\phi) \rightarrow \neg\mathcal{B}_r\phi^c \wedge \mathcal{B}_r\phi \wedge \mathcal{D}_r\mathcal{B}_s\mathcal{B}_r\phi) \\ & \quad \vee(\neg \mathit{bel\_revise}(\phi) \rightarrow \mathcal{D}_r\mathcal{B}_s\phi^c) \\ \vee & \neg\mathcal{K}_r\phi \rightarrow \\ & \quad (\mathit{consistent}(\phi) \rightarrow \mathcal{B}_r\phi \wedge \mathcal{D}_r\mathcal{B}_s\mathcal{B}_r\phi) \\ & \quad \vee(\neg \mathit{consistent}(\phi) \rightarrow \mathcal{B}_r\phi^c \wedge \mathcal{D}_r\mathcal{B}_s\phi^c) \end{array}$$

Note here that the formula ϕ^c denotes the complement of formula ϕ. This specification treats each of the three cases: when the content of the inform is already known, contradictory, and new. The function *bel_revise* is a gloss on a's belief revision for contradictory statements (cf. [7]), and we are not saying how the agent ensures its database is consistent (but there are algorithms for doing this, e.g. forward chaining). Consistent is evaluated to true or false by adding a formula to a belief database and checking, as the function name implies, for consistency.

In particular, the interpretation of this specification follows our intuitive reading of $\models [r, A]\phi \rightarrow \psi$ mentioned above: after the action, if ϕ is true, then try to make ψ true. The agent's intentional state is really being treated as a kind of active database, and in particular when dealing with the belief revision, the handling of $\neg\mathcal{B}_r\phi^c$ is not meant to add this negated formula, but to withdraw the existing belief.

Furthermore, this is just one formulation: the treatment of new and contradictory information may not be treated in the same way, for example. It is also easy to refine such a specification to incorporate elements of trust. Furthermore, since the sincerity condition is predicated on the notion of co-operation; and all speech acts involve the receiver recognizing an intention on the part of the sender, agents are free (but not forced) to make further inferences about the other agent's beliefs and intentions. Different inferences may be more appropriate for particular types of agents in different kinds of application.

Suppose we now encoded, for agent s, the axioms in its Plan Library as shown in Table 3. Note that these are plan axiom schema, and can be instantiated for any agent A and proposition ϕ. Then these axioms state, respectively that: firstly, if s believes some proposition ϕ, and wants (desires) that both itself and some other agent r believe to believe that proposition or its complement, i.e. they have a mutual belief in the truth- or falsehood of some proposition. If agent s believes that after successfully "doing" the *datasync* protocol it will achieve this goal, then it will form the intention to do the first action which will initiate that protocol (plan). Secondly, that after some agent informs it of a proposition ϕ using the dataysnc protocol, if it believes it and wants the sender to believe it, then it will form the intention to acknowledge it. Otherwise, if it believes the opposite, it will form the intention to inform the sender of that.

For the sending agent s, there are facts, such as p, q, etc. (implictly $\mathcal{B}_s p, \mathcal{B}_s q$, etc.) and more modal action schema. These states the beliefs that an agent will hold after performing or witnessing an action. For example, for agent s, we might have:

Belief Database: $\mathcal{B}_s$ p
$[A, \mathsf{acknowledge}(s, \phi, \_)]\mathcal{B}_A\phi$
$[s, \mathsf{acknowledge}(A, \phi, \_)]\mathcal{B}_A\mathcal{B}_s\phi$

These formulas are a belief in fact p, and axiom schemas that state after A acknowledges ϕ, A believes ϕ, and after s acknowledges ϕ, A believes that s believes ϕ. The post-condition of these modal action schemas specify the intended rational effects of the action which the agent *believes* will result after doing the action. (Note that these are still *beliefs* of s. They may not be the *actual* results, in which case the agent will have to do belief revision.) The formulas here concern the acknowledge performative, irrespective of the protocol it is used in. The first case is for when agent s is the receiver of the message, the second case for when it is the performer of the speech act.

In the desires database of s, there are goals. Now suppose that there is a desire for agent s that itself and some agent r have a mutual belief in p Then the combination of beliefs and desires is now used to `trigger` a plan, or protocol. In this case, it is the intention to `execute_one` act, which will initiate the *datasync* protocol, in effect a joint plan with r. This is achieved by sending the first message which initiates the protocol, in this case an inform. Thus, in this case, for s we get:

Intentions: $\mathcal{I}_s \ll s, \mathsf{inform}(r, p, \mathit{datasync})\gg$

Furthermore, now suppose agent r has identical axioms in its plan library, except substituting r for s. The belief database of agent r, the intended recipient of the speech act, might be entirely empty, except that it will have its versions of the axiom schema in its belief database and plan library, viz:

Beliefs: $\mathcal{B}_r$ $[A, \mathsf{acknowledge}(r, \phi, \_)]\mathcal{B}_A\phi$
$[r, \mathsf{acknowledge}(A, \phi, \_)]\mathcal{B}_A\mathcal{B}_r\phi$

and similarly for the axioms in its Plan Library.

After it receives the incoming message, r runs its procedure `process_message` on the content. If the specification above is mapped into Prolog, according to the mapping suggested earlier, resulting in the following code:

```
process_message( S, inform(R,P,datasync) ) :-
    believes( r, P ),
    assert( intends( r, acknowledge(S,P,datasync) ).
process_message( S, inform(R,P,datasync) ) :-
    believes( r, not(P) ),
    belief_revise( P ) ->
      ( retract( believes( r, not(P))),
      assert( believes( r, P)),
      retract( desires( r, believes( s,
                                    believes(r, P)))) ) ;
      ( assert( desires( r, believes(s,not(P)) ))).
process_message( S, inform(R,P,datasync) ) :-
    \+ believes( r, P ),
    \+ believes( r, not(P) ),
    consistent( P ) ->
      ( assert( believes( r, P)) ,
      retract( desires( r, believes( s,
                                    believes(r, P)))) ) ;
      ( assert( believes( r not(P))),
      assert( desires( r, believes(s,not(P)))) ).
```

then, using clause 3, first disjunction, the new belief $\mathcal{B}_r p$ and goal $\mathcal{D}_r\mathcal{B}_s\mathcal{B}_r p$ are added (p is not present in the belief database and adding it is consistent). Then the axiom schema in the plan library which is triggered after receiving the event is the second one, triggering the intention to acknowledge the inform. Thus the following beliefs, desires and intentions are added:

Beliefs:	$\mathcal{B}_r$	p
Desires:	$\mathcal{D}_r$	$\mathcal{B}_s\mathcal{B}_r p$
Intentions:	$\mathcal{I}_r$	$\ll r, \mathsf{acknowledge}(A, \phi, \mathit{datasync})\gg$

Now we see that the meaning of a speech act (at the action level) is indeed an intention to reply with a valid performative in the context of the protocol. Furthermore, after r executes the intention, by sending the message, the axiom schema in its belief database $[r, \mathit{acknowledge}(A, \phi, \_)]$ is instantiated (with $A = s$ and $\phi = p$), with the conclusion $\mathcal{B}_s\mathcal{B}_r p$. This achieves the goal which is duly discharged.

When s receives this message, the other axiom schema (i.e. the schema in which s is the receiver) concerning acknowledge in its belief database is instantiated in turn, with the conclusion $\mathcal{B}_r p$. This achieves s's original goal, which is in turn discharged.

5.4 A Reference Implementation

If any instance of an axiom schema in the Plan Library becomes true, then this combination of beliefs and desires will trigger a protocol as an action plan. An instance of this protocol will be instantiated with the conversation respondent, a new unique conversation identifier (e.g. c_1471 in Figure 5 below), and possibly some maintenance goals (i.e. these are goals, which if they become untrue, will cause the interpreter

$$
\begin{array}{l}
\textit{Plan Library}_s \\
\mathcal{B}_s\phi \wedge \mathcal{D}_s((\mathcal{B}_s\phi \rightarrow \mathcal{B}_A\phi) \wedge (\mathcal{B}_s\phi^c \rightarrow \mathcal{B}_A\phi^c)) \rightarrow \mathcal{I}_s \ll s, \mathsf{inform}(A, p, \textit{datasync}) \gg) \\
\mathrm{DONE}(\ll A, \mathsf{inform}(s, \phi, \textit{datasync}) \gg) \wedge \mathcal{B}_s\phi \wedge \mathcal{D}_s\mathcal{B}_A\mathcal{B}_s\phi \rightarrow \mathcal{I}_s \ll s, \mathsf{acknowledge}(A, \phi, \textit{datasync}) \gg \\
\mathrm{DONE}(\ll A, \mathsf{inform}(s, \phi, \textit{datasync}) \gg) \wedge \mathcal{B}_s\phi^c \wedge \mathcal{D}_s\mathcal{B}_A\phi^c \rightarrow \mathcal{I}_s \ll s, \mathsf{inform}(A, \phi^c, \textit{datasync}) \gg
\end{array}
$$

Table 3: Plan Library for agent s (and for r, with $s = r$)

to abandon the plan/conversation). This situation is illustrated in Figure 5. (Note that it is the integer part of the conversation identifier (e.g. 1471) which is used in the *conv* function, the $c\_$ part is for human readability only.)

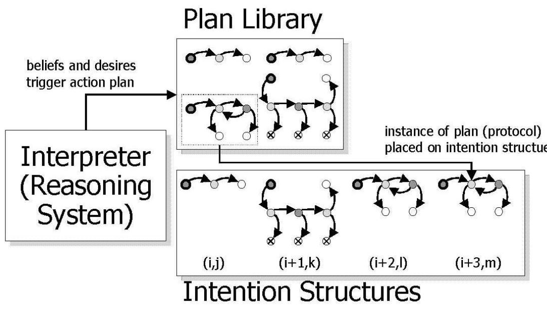

Plan Library
beliefs and desires trigger action plan
instance of plan (protocol) placed on intention structuers
Interpreter (Reasoning System)
(i,j)
(i+1,k)
(i+2,l)
(i+3,m)
Intention Structures

Figure 5: BDI Architecture with Protocols as Plans

Figure 5 also illustrates the idea that other conversations are going on at the same time as the new conversation is initiated and enacted. Note that not all these conversations may have been initiated by agent s, in some cases, other agents will have started a conversation with s. Each conversation is identified by a unique identifier, as descibed in [16]. These are standard parameters, for example, in the syntax of FIPA ACL [4].

Now, in the next round of the interpreter execution cycle, the agent may choose to perform one of the communicative acts available to it. Note that in some ongoing conversations, it may be waiting to receive a message (i.e. a speech act) from another agent. When such a message is received, the conversation identified by conversation identifier is advanced to the next state according to the performative and the protocol. Any conversation state variables are updated and the content of the message acted upon. These actions can all be specified and implemented as part of its *add* function. Note, however, that each agent could chose to implement things differently.

When an agent has a choice of possible performatives from the new protocol state, its *select* function determines which of these performatives is actually used. Again the *select* function may be different for different agents. It is only necessary to comply with the protocol description. Whatever way two agents implement the intentional specification (even without intentional modalities or explicitly represented beliefs, desires, etc.), then the specification of the protocol semantics should ensure that they can interoperate at the action level. Of course, there has also to be common agreement on syntax, ontology, etc., for interoperability at the content level, and this is another matter entirely.

6. SUMMARY AND CONCLUSIONS

The raison d'etre for this work was concern for standards in agent communication. In particular, the proposed FIPA ACL standard, while well-found for its application domain (co-operative answers to database queries) could not, we argue, serve as a standard for all agent applications [19].

Most of all, we found fault with the Gricean interpretation of speech acts as the basis the semantics. Instead, we took our own motivation from Natural Language understanding, using ideas from Montague Grammar [13], Discourse Representation Theory [8], and Conversation Analysis [12], and a typical Computer Science interpretation of Wittgenstein's meaning-as-use. From this, we developed a new semantic framework for describing the semantics of a *class* of ACLs, not just one, based on protocols.

Although this described a standard ACL semantics at a high level of abstraction, and therefore only focused on external behaviour, we also showed how deseigners were enabled to develop their own intentional and application-specific specifications for internal behaviour. This is then a very general and powerful development method, for which compliance to a standard is more straighforward to verify. Furthermore, these specifications may be made open (public), allowing third party designers to deploy systems that can interact with other agents, and comply with both the standard and the application requirements.

Singh [23] has articulated the properties that a standard ACL should satisfy, and this proposal should be evaluated against those requirements. However, the generality of our method has been substantiated by its application to four non-trivial examples: FIPA'97 ACL [15], KQML brokerage protocols and an auction protocol [16], and sACL, as described in this work and in [17]. In particular, for this last language we gave a reference architecture in terms of the BDI architecture, which can provide the basis of implementation. Ideally we would support this method with tools and agent engines.

The entire package is then precisely what good Software Engineering for multi-agent systems should be about: a comprehensible semantics, a general development method, a formal specification language, a reference architecture and tool support, preferably underpinned by acceptance as an international standard. This work has been presented at FIPA meetings, and while that institution has definitely moved in the direction of protocols (the use of UAML is especially commended, although even then at some point intentional specifications will still be required), it prefers to retain the 'legacy system' of FIPA'97 semantics.

We are taking this work in other directions. One avenue of research is the possibility that the exchange of intentional specifications can make a contribution to solving the more general ontology problem for open agent-based systems. The second avenue, being actively investigated in the new EC IST project ALFEBIITE, is the normative, legal, ethical and contractual issues underpinning agent communication in multi-agent systems. This investigation is possible precisely because our framework considers context as part of understanding and is not prematurely committed to any single intentional model of communication.

Acknowledgements

This work has been undertaken in the context of the UK-EPSRC/Nortel Networks funded project CASBAh (grant Ref. No. GR/L34440), and the support from these funding bodies is gratefully acknowledged. The constructive input of colleagues, in particular Abe Mamdani and Frank Guerin, is also noted.

We are also very appreciative of the many useful and detailed reviewer's comments, which were especially useful in revising the paper, and for the excellent editorial support provided by the workshop organizers.

7. REFERENCES

[1] C. Castelfranchi. Commitments: From individual intentions to groups and organizations. In V. Lesser, editor, *Proceedings ICMAS-95*. AAAI-MIT Press, 1995.

[2] P. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

[3] 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.

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

[5] M. Fisher. Representing and executing agent-based systems. In *Proceedings ECAI Workshop on Agent Theories, Architectures and Languages*, pages 307–323. 1994.

[6] M. Fisher. Implementing BDI-like systems by direct execution. In *Proceedings 15th IJCAI'97*, pages 316–321. 1997.

[7] J. Galliers. Autonomous belief revision and communication. In P. Gardenfors, editor, *Belief Revision*. Cambridge University Press, 1992.

[8] F. Guenthner. From sentences to discourse: Some aspects of the computational treatment of language. In A. Blaser, editor, *Natural Language at the Computer*, volume 320 of *LNCS*, pages 147–165. Springer-Verlag, 1988.

[9] D. Kinny, M. Georgeff, J. Bailey, D. Kemp, and K. Ramamohanarao. Active databases and agent systems: A comparison. In *Proceedings Second International Rules in Database Systems Workshop*. 1995.

[10] Y. Labrou and T. Finin. Semantics and conversations for an agent communication language. In M. Huhns and M. Singh, editors, *Readings in Agents*, pages 235–242. Morgan Kaufmann, 1998.

[11] Y. Labrou and T. Finin. Semantics for an agent communication language. In M. Singh, A. Rao, and M. Wooldridge, editors, *Intelligent Agents IV*, volume 1365 of *LNAI*. Springer-Verlag, 1998.

[12] P. Luff, N. Gilbert, and D. Frohlich, editors. *Computers and Conversation*. Academic Press, 1990.

[13] R. Montague, editor. *Formal Philosophy*. Yale University Press, 1974.

[14] J. Odell, H. v. d. Parunak, and B. Bauer. Representing agent interaction protocols in uml. In *Proceedings Autonomous Agents 2000*. to appear (2000).

[15] J. Pitt and F. Bellifemine. A protocol-based semantics for FIPA'97 ACL and its implementation in JADE. In E. Lamma and P. Mello, editors, *Proceedings 6th AI*IA Congress*. Editrice Pitagora, 1999.

[16] J. Pitt, F. Guerin, and C. Stergiou. Protocols and and intentional specifications of multi-party agent conversations for brokerage and auctions. In *Proceedings Autonomous Agents 2000*. to appear (2000).

[17] 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.

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

[19] J. Pitt and A. Mamdani. Some remarks on the semantics of FIPA's agent communication language. *Autonomous Agents and Multi-Agent Systems*, 1999.

[20] A. Rao and M. Georgeff. Modeling rational agents within a BDI-architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings 2nd International Conference on Principles of Knowledge Representation and Reasoning*. Margan Kaufmann, 1991.

[21] A. Rao and M. Georgeff. BDI agents: From theory to practice. In V. Lesser, editor, *Proceedings ICMAS95*. AAAI Press, 1995.

[22] D. Sadek. A study in the logic of intention. In *Proceedings 3rd Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 462–473. Cambridge, MA, 1992.

[23] M. Singh. Agent communication languages: Rethinking the principles. *IEEE Computer*, pages 40–47, 1998.