Using Distributed Protocols as an Implementation of Dialogue Games

Jarred P. McGinnis, David Robertson, and Chris Walton

Centre for Intelligent Systems and their Applications
School of Informatics
University of Edinburgh
Appleton Tower, Room 3.07
11 Crichton Street
Edinburgh EH8 9LE
(0)131 651-4156
j.p.mcginnis@sms.ed.ac.uk

Abstract. The dialogue game formalism presented in [MP02] can be implemented using a distributed interaction protocol [Rob03]. This provides a means for generating atomic dialogue types as well as the ability to form complex dialogue structures.

The agent's use of dialogue games is expressed in a shared distributed protocol language. Other agents are not required to know the intricacies of the dialogue games, but instead need only to understand the protocol language. All the rules necessary to play the dialogue game or follow the game framework are provided to the agent as a protocol passed between the two agents. The protocol language does this by adapting aspects of Electronic Institutions [ERAA+00] to express the agent dialogue as a distributed protocol, while avoiding some of the shortcomings of the EI approach (e.g. static definition of protocols and centralised control).

1 Introduction

Dialogue game frameworks attempt to construct more complex and robust agent conversations. This is achieved by combining different atomic dialogue types which have been identified by philosophers analysing human dialogues [WK95]. This approach tries to avoid the semantic ambiguities inherit in mentalistic models and the rigidity of static protocol-based approaches [FIP01]. The dialogue game approach depends on several assumptions about participating agents. Agents participating in the dialogue game framework must agree on all the rules of the framework. If a means to implement the formal definitions of the framework exists and it is possible to guarantee that the agents playing the dialogue game share the same rules, then this is an ideal approach to agent systems. However, in large open multi-agent systems it is not be practical to make these constraints or guarantees. Even though the Electronic Institution model [ERS+01] is limited and inflexible, it is easy to deploy and ensures that agents behave correctly within the system. The protocol language, described in this paper, utilises

the advantages of both dialogue games and Electronic Institutions. It has the flexible and dynamic nature of dialogue games and uses components of the well developed implementation of Electronic Institutions. The dialogue game framework can be written in the protocol language and then be distributed to the agent's communicative partners. This distributed protocol can then be followed by the other agents without needing to know anything about the dialogue game framework that generated it.

The remainder of this section provides background to the Dialogue Game framework. Section 2 discusses the framework, and section 3 explains the protocol language and its relationship to the framework. Lastly, Section 4 will provide an example and the final section will discuss further work and potential hazards related to this approach.

1.1 Dialogue Typology

The philosophers Doug Walton and Erik Krabbe have developed a typology of dialogues to detect fallacious reasoning [WK95]. This typology was adopted by Chris Reed [Ree98] in a formalism for multi-agent systems and inter-agent communication. Of the six kinds of dialogue identified, five of these dialogue types are applicable to the domain of agent communication. The sixth, eristic, is a dialogue where reasoning has ceased and the participants use the dialogue for the airing of grievances and one-upmanship. This dialogue type is important for the study of human conversations, but it is ignored by the agent research community. Dialogues are classified into the different types by three criteria. The first criteria considers the initial situation. What information do each of the participants have? Are the agents cooperative or competitive with each other? The second criteria concerns the individual goals an agent has for the interaction, and the third criteria are the goals shared by the participating agents. In *Information-Seeking* dialogues, one agent seeks the answer to a question which it believes the other agent possesses. *Inquiry* dialogues occur when two agents work together to find the answer to a question whose solution eludes both agents. A *Persuasion* dialogue has one agent attempting to convince another to adopt some proposition which it currently does not believe. *Negotiation* dialogues occur when the participants haggle over the division of a scarce resource. In *Deliberation* dialogues, the agents attempt to agree on a course of action for a particular situation. It is rare that any actual dialogue will be purely of one instance of one kind of dialogue. It is more likely that a dialogue will consist of an amalgamation of the different types. For example, during a negotiation, propositions may need clarification and an information-seeking dialogue would occur. This dialogue typology is fundamental to recent agent communicative models using dialogue games.

1.2 Dialogue Games

Dialogue games have existed for thousands of years, since Aristotle, as a tool for philosophers to formalise argumentation. This was an attempt to identify

when an argument or its justification is weakened or undercut by an argument or refutation made be the other participant. By each player making 'moves' and following a set of rules, it was hoped that properties of good and bad arguments could be identified. This formalism for argumentation has been employed to increase the interoperability of software agents, the objective being to produce a meaningful interaction between dialogical partners by following the rules of an individual dialogue game.

There are several components to a dialogue game. The commencement and termination rules specify the conditions under which a dialogue can start or end. The participants must share a set of locutions. This is a set of performatives from an agent communication language that is shared between the agents. This language must include the ability to utter assertions as well as justifications and challenges to those assertions. Another component is the combination rules. These rules define when particular illocutions are permitted, required, or illegal. The last part necessary for a dialogue game is the rules for commitment. These rules create obligations on the agent with respect to the dialogical moves of the agent. These commitments can be divided into dialogical and semantic. Dialogical commitments are the obligation of an agent to make a particular move within the context of the dialogue game. Semantic commitments indenture the agent to an action beyond the dialogue game itself. A record of these commitments is publicly stored. For example, if you say you are willing to pay the highest price in an auction, it will be known that you are committed to actually pay that price.

2 McBurney/Parsons Dialogue Framework

The formalism creates three tiers for agent dialogues. In order from lowest to highest, the layers are: topic, dialogue, and control. The topic layer is concerned with the domain of discourse being discussed in the dialogue, e.g. the topic of 'Beatles songs' or the topic of 'learning systems'. The remaining two layers are of more interest. The dialogue layer is the instantiation of the different types of dialogues, and the control layer is where agents decide which dialogues to take part in. This particular framework assumes that the agents willingly participate in any occurrence of a dialogue.

2.1 Dialogue Layer

This is the layer where the individual dialogue types occur. Each agent will have combination rules and commitment rules which will manifest themselves at this layer. These rules constrain the moves the participants can make at a specific point in the dialogue. The utterance of each illocution is dependant on the activity of the conversation before it. For example, a justification cannot be made until an assertion has been stated, or if an agent makes an assertion it cannot contradict that assertion without some retraction occurring. In this layer, there must be illocutions that make it possible for the agents to interrupt the dialogue

and return to the control layer. This illocution would be expressed if any commencement, termination, combination, or commitment rule which necessitates either the closing or the initiation of a dialogue. The actual beginning and ending of a dialogue type is done at the control layer.

2.2 Control Layer

This layer enables the agent to propose and agree to the specific dialogue types it wishes to engage in at the dialogue layer. Rules at this level define the legal combinations of the individual dialogue types. This is done by the expression of the illocution *BEGIN(G(p))* or *END(G(p))*. G is the type of dialogue to start or end and p is the topic of the dialogue game. The legal dialogues types are iteration, sequencing, parallelisation, embedding, and testing. *Iteration* is a finite number of one type of dialogue. Each occurrence of the dialogue begins directly after the previous one ends. *Sequencing* is similar to iteration but the next dialogue to begin can be of any type. *Parallelisation* represents the simultaneous expression of two dialogue types until each are closed. *Embedded* dialogues happen when during the course of a dialogue another is begun without ending the first. The embedded dialogue continues until it finishes, and the initial dialogue is continued at the point of interruption. This allowance for combinations of dialogues creates a difficulty for the rules for commitment, specifically semantic commitments. If a dialogue creates a semantic obligation and the commitment remains unsatisfied when another dialogue is opened, opinions differ on the ordering of satisfaction of the commitments. When an agent wants to suggest a dialogue sequence it is written like this, *BEGIN(G(p)∩H(p))*. This states that the agent would like to begin, in parallel ∩, the dialogues G and H about topic p. The ordering of the satisfaction of semantic commitments in the dialogues is written like this, *BEGIN(G(p)∩H(p) | SC(G(p)) > SC(H(P)))*, which states that the semantic commitments of G override those of H

3 The Distributed Protocol Language

The development of the protocol language is a reaction to work done on Electronic Institutions. Electronic Institutions provide structure to large and open multi-agent systems. By emulating human organisations, Electronic Institutions provide a framework which can increase interoperability. The EI regulates the agent's communicative activities by forcing agents to adhere to their roles, commitments, and obligations. Also, the EI ensures each agent participating within the system shares the same world-view about the conversation and its topics. Although the EI framework provides structure and stability to an agent system, it comes at a cost. Integral to EI is the notion of the administrative agents. The administrative agents' task is to enforce the conventions of the Institution and shepherd the participating agents. Messages sent by agents are sent through the EI. This synchronises the conversation between the conversing agents, and keeps the administrative agent informed of the state of the interaction.

An unreliable keystone makes the whole of the arch defective, just as the system is now dependent on the reliability and robustness of its administrative agent. Also, this centralisation of control runs counter to the agent paradigm of distributed processing. Within the scenes of Electronic Institutions, interaction protocols are defined to guarantee that agents utter the proper illocutions and utter them at the appropriate time. This is defined formally by the specifications of the EI and left to the designers of individual agents to implement. It assumes that the agent's interaction protocol covers the entire conversation space (all possible paths the dialogue may take) before the conversation occurs. If the interaction needs of the institution change, this would require redefinition of the Institution and re-synthesis of the individual agents. Agents are also expected to know the global state of the system and their exact position within it. In EIs this is handled by an administrative agent whose job it is to synchronise the multitude of agents involved.

The protocol language is an attempt to address some of these shortcomings of EIs but retain the benefits of implementing the EI framework. Its goal is to lessen the reliance on centralised agents for synchronisation of individual participants in the system, provide a means for dissemination of the interaction protocol and the separate the interaction protocol from the agent's rationalisations to allow the dynamic construction of protocols during the interaction. By defining interaction protocols during run-time, agents are able to interact in systems where it is impossible or impractical to define the protocol beforehand. The protocol defined in Figure 1 is similar to the protocol described in [WR02]. This technical paper uses the term 'Flexible Protocol'. The protocol alone is not particularly flexible, but it is the use of the protocols and the protocol language that enables flexibility.

The syntax of the protocol language is shown in Figure 1. An agent protocol is defined as an agent definition and composed with an operation. The agent definition individuates the agents participating in the conversation(id), the role the agent is playing(r) and the scene in which the role is being played(n). Operations can be classified in three ways: actions, control flow, and conditionals. Actions are the sending or receiving of messages, a no op, or the adoption of a role. Control Flow operations temporally order the individual actions. Actions can be put in sequence (one action must occur before the other), parallel (both action must occur before any further action), or choice (one and only one action should occur before any further action). Conditionals are the preconditions and postconditions for operations. The message passed between two agents using the protocol consists of three parts. The first is the actual illocution the agent is wishing to express. The second is the full protocol itself. This is the protocol for all agents and roles involved in the conversation. This will be necessary for the dissemination of the protocol to agents. The last part to be sent is a copy of the dialogue as it occurs, the dialogue clause. The dialogue clause serves as a history of the conversation as it progresses, a marker for the current state of the dialogue, and a record of the values given to variables. Other features of the protocol are the inclusion of constraints on the dialogue and the use of roles.

$$
\begin{array}{lll}
P & \in \text{Agent Protocol} & ::= \theta :: op. \\
\theta & \in \text{Agent Definition} & ::= \mathbf{agent}(id, r, n) \\
op & \in \text{Operation} & ::= \text{no op} \\
& & \mid \quad \theta \\
& & \mid \quad (op) \qquad\qquad\qquad \text{(Precedence)} \\
& & \mid \quad \rho \Rightarrow \theta \qquad\qquad\quad \text{(Send)} \\
& & \mid \quad \rho \Leftarrow \theta \qquad\qquad\quad \text{(Receive)} \\
& & \mid \quad op1 \ \mathbf{then} \ op2 \qquad \text{(Sequence)} \\
& & \mid \quad op1 \ \mathbf{or} \ op2 \qquad\quad \text{(Choice)} \\
& & \mid \quad op1 \ \mathbf{par} \ op2 \qquad \text{(Parallelism)} \\
& & \mid \quad op \rightarrow \psi \qquad\qquad \text{(Prerequisite)} \\
& & \mid \quad op \leftarrow \psi \qquad\qquad \text{(Consequence)} \\
S & \in \text{Scene} & ::= n[R, A, P] \\
\rho & \in \text{performative} & \\
\psi & \in \text{state} & ::= \text{a procedural call} \\
& & \quad \text{to express a precondition or} \\
& & \quad \text{postcondition of an operation}
\end{array}
$$

Fig. 1. The Definition of the Protocol

An agent's activities within a multi-agent system are not determined solely by the agent, rather it is the relationship to other agents and the systems itself that helps determine what performative an agent will utter. These can be codified as roles. This helps govern the activity of groups of agents rather than each agent individually. Constraints are marked by '←' (requirements) and '→' (consequences). These are requirements or consequences for an agent on the performatives or roles available to it. The constraints are intended to provide the agent with a shared semantic for the dialogue. These constraints communicate meaning and implication of the action to the agent's communicating partner. For example, an agent receiving a protocol with the constraint to believe a proposition p upon being informed of p can infer that the agent sending the protocol has a particular semantic interpretation of the act of informing other agents of propositions. '⇐' and '⇒' mark messages being sent and received. On the left-hand side of the double arrow is the message and on the right-hand side is the other agent involved in the interaction. '⇐' is a message being received and a '⇒' is a message being sent. After any operation a constraint can be placed upon that operation.

3.1 Dialogue Game as Protocol

If the dialogue game framework is going to be implemented, agents need to be able to automatically generate the dialogue types. There is speculation on how this may be accomplished for each of the dialogue types. It is certainly possible to formulate an idealisation of each of the individual dialogue types. In an ideal *information-seeking* dialogue one agent receives a request for information, and

the agent immediately responds with the requested information. The task then is to ensure that at any point in this dialogue it will be possible to include any of the other types of dialogue. In the example shown in section 4, the combination types are explicitly encoded into the protocol. The dialogue types are encoded as roles. This creates a modular approach which is in accordance with the spirit of dialogue game framework. Each important aspect of the dialogue game framework can be addressed by an implementation in the protocol. This will allow an agent to communicate with other agents that have not been designed with dialogue games as their communicative model. The rules of the dialogue game and framework are expressed as a protocol. The agent receiving protocol will be able to communicate by following the protocol. It does not need to know that a message it sends follows a certain dialogue game rule. It is only required to know that the message it sends is appropriate according to the protocol it has received.

3.2 Components of the Dialogue Game

The five components of a dialogue game are commencement rules, illocutions, combination rules, commitment rules, and termination rules. All of these components can be expressed in the protocol language. However, it will still be necessary for the agents to agree upon a set of illocutions. These rules, expressed as protocol and sent to an agent to adopt, could be construed as in impediment to an agent's autonomy. The protocol and the constraints of the protocol do not dictate any of the agent's deliberative processes. The constraints are merely a means to ensure the agent interaction is consistent. By their interaction in a multi-agent system, agents willingly sacrifice autonomy in order to gain utility by following the norms and mores of the system in which it is participating. Whether these norms are followed by the agent being hard coded by an engineer interpreting a specification or by the agent obeying the constraints of the protocol makes little difference.

3.3 Layers of the Framework

The three layers introduced in the dialogue game framework are the topic layer, the dialogue layer, and the control layer. The dialogue layer still exists and appears as the dialogue clause as the dialogue progresses. The control layer is no longer necessary. The protocol can be designed so as to allow an agent to refuse to take up some role in the dialogue.

4 An Example

Two agents, Joe and Doctor Doe, are having a conversation. Joe has a lump, and asks the doctor what it might be. The diagnosis scene is a good example where it would be difficult to map out the entire conversation as done with the EI approach. This interaction would be better suited to a more flexible model of

agent communication such as the dialogue game discussed previously. Before the doctor agent can provide the answer to Joe's question, the doctor agent will have a question of its own. Once Joe provides a response to the doctor's question, the doctor can then provide the answer to Joe's question. This is an example of two instances of an information-seeking dialogue. The first instance is the Joe agent seeking the identification of the lump. The second is the doctor agent seeking to know Joe's other symptoms, and it is embedded within the first.

4.1 The Example using Mcburney/Parsons Framework

---

1)Joe:BEGIN(INFOSEEK(Diagnosis))
 The agent requests the commencement of an information-seeking type dialogue about the topic of diagnosis
2)Doctor Doe:AGREE(INFOSEEK(Diagnosis))
 The agent Doctor Doe agrees.
INFORMATION-SEEKING Dialogue 1 opens.
3)Joe:REQUEST(lump)
Joe requests the identity of its lump from the Doctor Doe
4)Doctor Doe:PROPOSE-RETURN-CONTROL
Return to CONTROL Layer.
5)Joe:AGREE(RETURN-CONTROL)
6)Doctor Doe:BEGIN(INFOSEEK(Symptoms))
The agent Doctor Doe requests the start of information-seeking dialogue concerning the topic of symptoms.
7)Joe:AGREE(INFOSEEK(Symptoms))
INFORMATION-SEEKING Dialogue 2 opens, embedded in 1.
8)Doctor Doe:REQUEST(other_symptoms)
 Doctor Doe asks Joe for other symptoms.
9)Joe:INFORM(other_symptoms = itchy,redness)
Dialogue 2 concludes and the conversation returns to the initial info-seeking dialogue
10)Doctor Doe:INFORM(lump = bug_bite)
With this information the agent Doctor Doe can now provide the information that Joe was seeking.

Fig. 2. The Dialogue Game Framework in Action

An English interpretation will follow certain steps in the dialogue game for clarity. I have attempted to emulate the style of the example used in [MP02]. Each numbered line in Figure 2 identifies a message passed between the two agents. Lines 3, 8, 9, and 10 are illocutionary messages and are exactly the same messages that will be passed between the agents using the protocol. Lines 1 and

2 and Lines 6 and 7 are examples of the conversation at the control level of the dialogue game framework. There is no line explicitly ending any dialogue with a *END* message. This is only necessary if the dialogue is ended prematurely. One agent suggests a conversation type and topic to undertake and the other agent, in this example, agrees. This is unnecessary when using the protocol. It is not assumed that the agent knows about the dialogue types and an agent who wishes to begin a particular type of dialogue merely makes the first move in that dialogue type and provides the communicative partner with a copy of the protocol. This does not mean that the other agent is required to participate in the dialogue. Lines 4 and 5 in the example show one agent interrupting a dialogue instance to begin another. This is done by one agent proposing to return to the control layer. When the protocol is used the combination of dialogue types no longer has to be proposed. The protocol is designed to explicitly encode these combinations. A dialogue combination is made simply by taking that route in the protocol. Whether that agent recognises the move as a dialogue type combination is left to the designer. Figure 3 illustrates the protocol for a simple information-seeking dialogue. It allows for two types of dialogue combinations embedding and iteration. With this protocol it is possible to generate the example in Figure 2. The agents will initially have the role stated in line one. The role of being part of an Information seeking dialogue about a proposition p which is related to some topic. If an agent receives a request for information about p from an agent also playing an *Infoseek* role, it follows the protocol branch of line two which has the agent taking the role of an *Infoprovider*. If the agent satisfies the constraint on line 4, i.e. it needs to know something about p, it sends a request about p and then takes the role of an *Infoseeker*. After an instance of an information-seeking dialogue is completed, the agent can start another by retaking the role of *Infoseek* stated in line 6. The *Infoprovider* role is defined at line 7. If the agent knows p, it sends an inform message to the agent that sent the request. If the agent does not know p, it then will attempt to redress this by initiating another instance of an Infoseek (line 9). After which the agent will retake the role of *Infoprovider* with respect to the original topic and request for information. It may be that several iterations of this may occur. For example, Joe's reply of symptoms could lead the Doctor to request more information for purposes of clarification, and it would be necessary to ask another question before an answer to the identity of the lump could be reached. Once the inform is sent, the *Infoprovider* role is finished and the agent would continue through the protocol which would be line 6 in the *Infoseek* role. The *Infoseeker* role is similar to the *Infoprovider* role. On line 12. The receiving of the inform has a consequence of *assert(p)*. This states that p is now known by the agent and it becomes common knowledge the the action of assert. This could be used as an implementation of the shared commitment store briefly discussed in [MP02].

The dialogue clauses of Figure 4 and Figure 5 are the actual branches the agents take in the conversation by utilising the protocol defined in Figure 3. Figure 4 is the dialogue clause for the agent Joe who is seeking to know the identity of its lump. Figure 5 is the dialogue clause for the agent Doctor Doe,

```
1)agent(Infoseek(Topic,P),Id)::=
2)    (request(P) ⇐ agent(Infoseek(Topic,P),Cid) then
3)        agent(Infoprovider(Topic,P),Id)) or
4)    (request(P) ⇒ agent(Infoseek(Topic,P),Id) ← needtoknow(p) then
5)        agent(Infoseeker(Topic,P),Id)) then
6)agent(Infoseek(Topic',Q),Id).

7)agent(Infoprovider(Topic,P),Id)::=
8)    inform(P) ⇒ agent(Infoseeker(P),Cid) ← know(P) or
9)    (agent(Infoseek(Topic",R))then
10)        agent(Infoprovider(Topic,P),Id)).

11)agent(Infoseeker(Topic,P),Id)::=
12)    inform(P) ⇐ agent(Infoprovider(Topic,P),Pid) → assert(P) or
13)    (request(_) ⇐ agent(Infoseeker(_,_),Pid) then
14)        agent(Infoseek(Topic,_)) then agent(Infoseeker(Topic,P),Id)).
```

Fig. 3. Information-seeking protocol

who before it can identify Joe's lump, it need to know its other symptoms. As the conversation routes are explored the dialogue clause is appended. Each new step added to the dialogue clause is the actual step taken by the agents following the protocol that is passed between them.

```
agent(Infoseek(Diagnosis,lump),Joe) ::=
    request(lump) ⇒ agent(Infoseek(Diagnosis,lump),Doctor Doe) then
    agent(Infoseeker(Diagnosis,lump),Joe) ::=
        request(other_symptoms) ⇐
            agent(Infoseek(Symptoms,other_symptoms),Doctor Doe) then
        agent(Infoseek(Symptoms,other_symptoms),Joe) ::=
        agent(Infoprovider(Symptoms,other_symptoms),Joe) ::=
            inform(other_symptoms = itchy, redness) ⇒
    agent(Infoseeker(Symptoms,other_symptoms),Doctor Doe) then
        agent(Infoseeker(Diagnosis,lump),Joe) ::=
            inform(lump=bug_bite) ⇐ agent(Infoprovider(Diagnosis,lump)).
```

Fig. 4. Doctor-patient dialogue clause(Joe)

```
agent(Infoseek(Diagnosis,lump),Doctor Doe) ::=
    request(lump) ⇐ agent(Infoseek(Diagnosis,lump),Joe) then
    agent(Infoprovider(Diagnosis,lump),Doctor Doe) ::=
    agent(Infoseek(Symptoms,other_symptoms),Doctor Doe) ::=
        request(other_symptoms) ⇒
            agent(Infoseeker(Symptoms,other_symptoms),Joe) then
        agent(Infoseeker(Symptoms,other_symptoms),Doctor Doe) ::=
            inform(other_symptoms = itchy, redness) ⇐
                agent(Infoprovider(Symptoms,other_symptoms),Doctor Doe) then
            agent(Infoprovider(Diagnosis,lump),Doctor Doe) ::=
                inform(lump=bug_bite) ⇒ agent(Infoseeker(Diagnosis,lump)).
```

Fig. 5. Doctor-patient dialogue clause(Doctor Doe)

5 Conclusions and Observations

The use of the protocol is not seen as a replacement for either the dialogue game approach or Electronic Institutions. Its purpose is to bridge the implementation gap which exists. Dialogue Game frameworks, though rich in formal definition, lack a practical implementation. Electronic Institutions are an excellent model for implementation but its static state-based approach to agent conversations limit it to simple domains with small conversation spaces (e.g. auctions). By mediating between the two, the protocol described in the paper gains benefits from both approaches. The protocol language borrows several concepts from Electronic Institutions. EIs, through implementations and formal definitions, have been shown to be a solid approach to open multiagents systems. The protocol's distributed nature and run-time execution allows it to avoid some of the shortcomings of EIs. Encoding the dialogue game framework into a protocol allows an agent to use a very expressive model of agent communication without the drawback of depending on other agent designers to fully comply with the numerous(i.e. commencement, termination, combination, commitment) rules associated.

The protocols usefulness is not limited to implementing dialogue games or the McBurney/Parsons framework. A small scheduling program has been developed using the protocol written in Prolog and using LINDA. A Java-based agent framework also exists which uses an XML representation of the protocol. Separating the protocol from the deliberative and communicative models of agency makes definition and verification simpler tasks. Tools have already been developed which use model-checking for automatic verification. Work is also under-way to create a full implementation of the ideas presented in this paper. It includes all the atomic types of dialogue as well as the various combinations considered in the formal framework. There are a few issues that still need to be addressed. Such as the concern that the volume of data being exchanged could be a hindrance to this approach. Implementations using the protocol have been

created and this particular concern has not been a problem. That is not to say there is no place for optimisation, such as determining the minimal message that can be sent between agents.

This paper has shown that the distributed protocol language described is expressive enough to represent one of the more popular communicative models of agent research. This is done in such a way that an agents could communicate by using the protocol and not relying on agent's to share that communicative model.

References

[ERAA⁺00] Marc Esteva, Juan A. Rodrguez-Aguilar, Josep Ll. Arcos, Carles Sierra, and Pere Garcia. Institutionalising open multi-agent systems. In *proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS'2000)*, pages 381–83, Boston, 2000. ICMAS.

[ERS⁺01] Marc Estava, J. A. Rodriguez, Carles Sierra, P. Garcia, and J.L. Arcos. On the formal specifications of electronic institutions. *LNAI*, pages 126–147, 2001.

[FIP01] FIPA. FIPA communicative act library specification, http://www.fipa.org/specs/fipa00037/XC00037H.html, 2001.

[MP02] Peter McBurney and Simon Parsons. Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of Logic, Language and Information*, 11(3):315–334, 2002.

[Ree98] Chris Reed. Dialogue frames in agent communication. In Y. Demazeau, editor, *Proceedings of the Third International Conference on Multi-Agent Systems(ICMAS-98)*, pages 246–253. IEEE Press, 1998.

[Rob03] Dave Robertson. Distributed agent protocols. Technical Report contact author for details(dr@inf.ed.ac.uk), University of Edinburgh, 2003.

[WK95] Doug Walton and Eric C. W. Krabbe. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning.* SUNY press, Albany, NY, USA, 1995.

[WR02] Chris Walton and Dave Robertson. Flexible multi-agent protocols. Technical Report EDI-INF-RR-0164, University of Edinburgh, 2002.