Dynamic and Distributed Interaction Protocols

Jarred McGinnis[⋆] and David Robertson[⋆]
[⋆]Centre for Intelligent Systems and their Applications
University of Edinburgh
Appleton Tower, Room 4.15
Edinburgh EH8 9LE
j.p.mcginnis@sms.ed.ac.uk

Abstract

This paper describes a protocol language which can provide agents with a flexible mechanism for coherent dialogues. The protocol language does not rely on centralised control or bias toward a particular model of agent communication. Agents can adapt the protocol and distribute it to dialogical partners during interactions.

1 Introduction

As the programming paradigm of agency evolves, more robust, diverse, and complex agents are developed. The growing heterogeneity of agent societies will increase even further as the research and development of deliberative and communicative models produce new and interesting approaches (Pasquier et al., 2003). The need for an equally adaptive means of communication between this heterogeneous multitude also grows.

Electronic Institutions (Estava et al., 2001) and other state-based approaches are not feasible for use in open multi-agent systems with dynamic or large conversation spaces. The term conversation space is used to express every possible sequence and combination of messages that can be passed between two or more agents participating in a given agent system. Protocols provide a useful framework for agent conversations and the concern that they sacrifice agent autonomy is exaggerated. In social interactions, humans and agents must willingly sacrifice autonomy to gain utility. If I want my train tickets or cup of coffee, I must follow the implicit protocol and join the queue. It is the same for software agents. If the agent must gain a resource only available by participating in an English auction, it behoves the agent to adopt the protocol necessary for participation in the auction. Whether this is done by an explicitly defined protocol or the agent learning the protocol implicitly makes no difference to the agent's behaviour within the system.

Electronic Institutions take a societal approach to agent communication. Control is top-down. Administrative agents perch above the system and keep an eye on the agents as they interact inside the system. Agent-centric approaches build systems bottom-up. These approaches attempt to pack individual agents with a model of communication which can react to a multi-agent system. They have a communicative model that sits besides or is intertwined with its rational model. This is done in a number of ways. The most common is a BDI-model of communication as typified by the standardisation organisation, FIPA. There is a lot of dissatisfaction with FIPA ACL, and a variety of alternative models for agent communication have been proposed. All trying to address faults perceived with the FIPA approach.

The protocol language described in this paper seeks a balanced approach. It utilises the useful aspects of Electronic Institutions without relying on administrative agents or statically defined protocol specifications. Agents communicate not only individual messages but the protocol and dialogue state as well. The use of protocols provides structure and reliability to agent dialogues. Yet, by describing protocols as a process rather than a fixed state-based model, the conversation space can be defined as the agent interaction progress rather than being statically defined during the engineering process. Distributing the protocol along with the message also allows agent to communicate the convention for communication as well as coordinate the dialogue.

Section 2 will discuss some of the dominant agent communication paradigms. Section 3 describes the syntax and features of the protocol language. A discussion of adaptable protocols in section 4 is followed by an example illustrating an adaptable protocol using dialogue games in section 5. Section 6 concludes the paper enumerating the accomplishments and potential issues associated with the approach.

2 Approaches to Agent Communication

2.1 Electronic Institutions

Electronic Institutions(EI) provide structure to large and open multi-agent systems(MAS). By emulating human organizations, Electronic Institutions provide a framework which can increase interoperability. The EI framework formally defines several aspects of an agent society. The core of an EI is the formal definition of roles for agents, a shared dialogical framework, the division of the Institution into a number of scenes and a performative structure which dictates, via a set of normative rules, the relationships between the scenes. Agents interact with an Institution through the exchange of illocutions, i.e. messages with intentional force.

Participating agents are required to adopt a role within the Institution. This is similiar to our entering a shop and assuming the role of a customer, and the employee adopting the role of salesperson. A role is defined as a finite set of dialogical actions. By the adoption of a role within an Institution, an agent's activities within the Institution can be anticipated. This abstraction of agents as a role allows the Institution to regulate and identify agent activities without analysing individual agents. Relationships between agents can be dealt with as generalizations. A role can be defined as subsuming or being mutually exclusive to another role.

The dialogical framework provides a standard for communication. Agents are guaranteed to have a shared vocabulary for communication as well as a common worldview with which to represent the world they are discussing. The dialogical framework is defined as a tuple consisting of an ontology, a representation language, a set of illocutions, and a communication language. The representation is an encoding of the knowledge represented by the ontology and makes up the inner language. This is contained with an individual illocution that is passed between agents. The illocution, as part of the outer language or communication language, expresses the intention of the agent by its communicating the message of the inner language. The dialogical framework, which contains the ontological elements, is necessary for the specification of scenes.

All interactions between agents occur within the context of scenes. Scenes are interaction protocols between agent roles. They are expressed as a well-defined protocol which maps out the conversation space between two agent roles. These scenes are represented as graphs. The nodes are conversation states and arcs representing the utterances of illocutions between the participants. Each scene will have a set of entrance and exit states with conditions that must be satisfied before the agent can begin or exit a scene. A set of roles and scene states are formally defined. An element of the set of states will be the initial state and a non-empty subset will be final states. Between the states there is a set of directed and labelled edges.

Scenes are individual agent conversations. In order for agents to participate in more interesting activities, it is necessary to formalize relationships between these individual conversations. The performative structure formalizes this network of scenes and their association with each other. The roles an agent adopts and the actions of the agents create obligations and restrictions upon the agent. These obligations restrict the further movement of agents. The performative structure is made of a finite non-empty set of scenes. There is a finite and non-empty set of transitions between these scenes. There is a root scene and an output scene. Arcs connect the scenes of the Institution. These arcs have different constraints placed upon them. For example, the constraints can synchronize the participating agents before the arc can be fully traversed, or there are constraints that provide an agent a choice point upon which scene to enter.

Within the scenes of an Electronic Institution, the actions an agent performs affect the future actions available to the agent. These consequences can extend beyond the current scene. These consequences could be the requirement for a agent to perform an action in some future scene or even which scenes or sequence of scenes an agent is now required to be a participant. These normative rules are categorized between two types. *Intra-scene* dictate actions for each agent role within a scene, and *inter-scene* are concerned with the commitments which extend beyond a particular scene and into the performative structure (Esteva et al., 2000).

Tools (Esteva et al., 2002) exist to aid in the creation of the various components and development of Electronic Institutions. This includes a tool to verify any specifications developed as well as tools to aid the synthesis of agents that can participate in the Electronic Institution (Vasconcelos, 2002).

2.2 Agent-centric Design

FIPA ACL for better or for worse has made a large impact of agent communication research. A victim of its own success, most new approaches to agent communication are attempts to redress FIPA's ACL deficiencies. Conversation Policies (Greaves et al., 2000) were an attempt to produce a more 'fine-grained' means of generating dialogues. More recently, researchers have developed communicative models to address the semantic verification problem of FIPA ACL (Wooldridge, 2000). There are other approaches which see the importance of separating the agent's internal states from the conversational model (Maudet and Chaib-draa, 2002). Two approaches of interest are theories based on social commitment or obligation and formal definitions of agent systems based on dialogue theory.

2.2.1 FIPA ACL

The Foundation for Intelligent Physical Agents develops software standards for agent communication. This is expressed in their official mission statement: *The promotion of technologies and interoperability specifications that facilitate the end-to-end interworking of intelligent agent systems in modern commercial and industrial settings.* In practice, this includes the publishing of standards concerning speech acts, predicate logic, and public ontologies. The individual communicative acts of FIPA's Agent Communication Language (ACL) is based on the speech act theory of Searle (1969). The semantics of FIPA ACL are based on the Belief-Desire-Intention (BDI) model of agency and is formalised in a language SL(for Intelligent Physical Agents, 2000).

Each communicative action by a FIPA compliant agent implies that agent is following the requirements specified for that action. This includes general properties for all communicative acts, the interaction protocol of which the act is a part, and the feasible preconditions and rational effects for that particular act (FIPA, 2001). For example, an agent *i* must believe a proposition *p* and believe that an agent *j* neither has any amount of belief about *p* or not *p* before it can send an **inform** FIPA ACL communicative act to agent *j*. Afterwards, agent *i* is entitled to believe that agent *j* believes *p*.

2.2.2 Social Commitment

Researchers have adopted the idea of social commitments to redress the semantic difficulties that arise when agents rely on mentalistic (e.g BDI) Agent Communication Languages (ACL). Social-based semantics consider the agent's relationship to its communicative partners. It is a recognition that an agent's communicative acts do not exist in a vacuum. It is the use of the intuitive idea that an agent's communication is an event which necessarily involves other agents.

Singh (2000) identifies several criteria for the semantics of an Agent Communication Language. According to Singh, an ACL should be formal, declarative, verifiable, and meaningful. To this end, he has developed a *social semantics*. He defines three facets to every communicative act. The *objective claim* which commits an agent to another that some proposition *p* holds. The *subjective claim* is that an agent believes *p*, and the *practical claim* that the agent has some justification or reason for believing *p*. This is a novel approach, because most reactions to the semantic verification problem of the mentalistic approach is to completely throw it away. Singh has, instead, embraced the mentalistic approach but coupled it with the idea of social commitment. The purely mentalistic approach rests on the assumption that the agent is sincere about *p*, but Singh has added that the agent is also socially committed to being sincere about *p*. It is recognized that the use of social semantics does not replace the need for protocols, but the combination of social semantics and protocols would create a much more flexible ACL Maudet and Chaib-draa (2002).

The approach described in Flores and Kremer (2002) uses the commitment themselves to develop the conversation between two agents. Flores argues that our verbal utterances carry with them obligations dependent on the role of the agent within a society. The question 'What time is it?' carries with it the obligation (in polite society) to not only reply but make an attempt to actually find out the time. The use of social commitments in multi-agent communication is to provide a number of rules that dictate appropriate illocutions and actions performed based on the agent voluntarily obligating itself to commitments with other agents and eventually discharging those commitments. A protocol is defined for the negotiation of the adoption of social commitments. Agents propose to add and remove commitments for action from personal commitment stores. An agent will propose to add a commitment to perform some actions. Once this is accepted and the commitment is satisfied the protocol includes steps to propose the release of any further commitment to that action. It is through this simple protocol and the social commitment-based conversation policies an agent conversation can be developed.

2.2.3 Dialogue Theory and Games

The philosophers Doug Walton and Erik Krabbe have developed a typology of dialogues to detect fallacious reasoning (Walton and Krabbe, 1995). This typology was adopted by Chris Reed (Reed, 1998) 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 criterion considers the initial situation. What information does each of the participants have? Are the agents cooperative or competitive with each other? The second criterion concerns the individual goals an agent has for the interaction, and the third criterion 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.

Dialogue games have existed for thousands of years, since Aristotle, as a tool for philosophers to formalise argumentation. It is 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 complexity and robustness of software agents conversations. The objective is 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. Firstly, the participants must share a set of locutions. This is a common requirement for models of agent communication. The commencement and termination rules specify the conditions under which a dialogue can start or end. 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.

Dialogue game frameworks (McBurney and Parsons, 2002; Maudet and Evrard, 1998) 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 (Walton and Krabbe, 1995). This approach avoids the semantic ambiguities inherent in mentalistic models and the rigidity of static protocol-based approaches (FIPA, 2001). 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. The number of requirements made on individual agents in order for them to play dialogue games makes the approach unsuited for open multi-agent systems.

3 The Protocol Language

The development of the protocol language is a reaction Electronic Institutions (Walton and Robertson, 2002). 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. Their 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 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 addresses some of these shortcomings of EIs but retains 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 language defined in Figure 1 is similar to the protocol language described in Walton (2004b) for which the formal semantics have been defined.

P	$\in$	Protocol	::	$\langle S,A^{\{n\}},K\rangle$
A	$\in$	Agent Clause	::	θ :: *op.*
θ	$\in$	Agent Definition	::	**agent**(*r*,*id*)
op	$\in$	Operation	::	null
			|	θ
		(Precedence)	|	(*op*)
		(Send)	|	$M \Rightarrow \theta$
		(Receive)	|	$M \Leftarrow \theta$
		(Sequence)	|	*op1* **then** *op2*
		(Choice)	|	*op1* **or** *op2*
		(Parallelism)	|	*op1* **par** *op2*
		(Consequence)	|	$\psi \leftarrow M \Leftarrow \theta$
		(Prerequisite)	|	$M \Rightarrow \theta \leftarrow \psi$
M	$\in$	message	::	$\langle m,P\rangle$
ψ	$\in$	state	::	a predicate

Figure 1: The abstract syntax of the protocol

Figure 1 defines the syntax of the protocol language. An agent protocol is composed of an agent definition and an operation. The agent definition individuates the agents participating in the conversation (*id*), and the role the agent is playing (*r*). 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), in parallel (both action must occur before any further action), or given a choice point (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 (*m*) the agent is wishing to express. The second is the full protocol (P) itself. This is the protocol for all agents and roles involved in the conversation. This will be necessary for the dissemination of the protocol as new agents enter the system. Other aspects of the protocol are the inclusion of constraints on the dialogue and the use of roles. 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 system itself that helps determine what message an agent will send. These can be codified as roles. This helps govern the activity of groups of agents rather than each agent individually. Constraints are marked by a '$\leftarrow$'. These are requirements or consequences for an agent on the occurrence of messages or the adoption of roles. The constraints provide the agent with a shared semantics 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 *s* upon being informed of *s* can infer that the agent sending the protocol has a particular semantic interpretation of the act of informing other agents of propositions. The '$\Leftarrow$' and '$\Rightarrow$' 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.

An agent must be able to understand the protocol, the dialogue state, and its role within the protocol. Agents need to be able to identify the agent clause which pertains to its function within the protocol and establish what actions it must take to continue the dialogue or what roles to adopt.

3.1 Implementing the Protocol Framework

A message is defined as the tuple, $\langle m,\mathrm{P}\rangle$. Where *m* is the message an agent is currently communicating, and P is the remainder is the protocol written using the language described in figure 1. The protocol, in turn, is a triple, $\langle S,A^{\{n\}},K\rangle$. *S* is the dialogue state. This is a record of the path of the dialogue through the conversation space and the current state of the dialogue for the agents. The second part is a set of agent clauses, $A^{\{n\}}$, necessary for the dialogue. The protocol also includes a set of axioms, *K*, consisting of common knowledge to be publicly known between the participants. The sending of the protocol with the messages allows agents to represent the various aspects of Electronic Institutions described in section 2.1. In addition, agents themselves communicate the conventions of the dialogue. This is accomplished by the participating agents satisfying two simple engineering requirements. Agents are required to share a dialogical framework. The same is required of Electronic Institutions, and is an unavoidable necessity in any meaningful agent communication. This includes the requirements on the individual messages are expressed in a ontology understood by the agents. The issue of ontology mapping is still open, and its discussion extends beyond the scope of this paper. The second requirement obligates the agent to provide a means to interpret the received message and its protocol. The agent must be able to unpack a received protocol, find the appropriate actions it may take, and update the dialogue state to reflect any actions it chooses to preform.

Figure 2 describes rule for expanding the received protocols. Details can be found in Robertson (a). A similiar language for web services is described in Robertson (b). An agent receives a message of the form specified in figure 1. The message is added to the set of messages, M_i, currently being considered by the agent. The agent takes the clause, C_i, from the set of agent clauses received as part of P. This clause provides the agent with its role in the dialogue. The agent then expands C_i by the application of the rules in figure 2. The expansion is done with respect to the different operators encountered in the protocol and the response to M_i. The result is a new dialogue state, C_n; a set of output messages, O_n and a subset of M_i, which is the remaining messages to be considered,M_n. The result is arrived at by applying the rewrite rules. The sequence would be similar to figure 3. C_n is then sent as part of P which will accompany the sending of each message in O_n.

3.2 Features of the Protocol

Several features of the protocol language are useful for agents capable of learning and adapting to the multi-agent system in which they participate. Sending the dialogue state during the interaction provides agents with several advantages. It is no longer necessary for an administrative agent to shepherd the interaction. The sending of the protocol with the message uses the 'hot potato' approach to communication. The interaction is coordinated by which agent currently 'holds' the protocol. The reception of a message would cue an agent to action. The sending of the protocol provides a means for dissemination of the social conventions for the dialogue. The most common approach is to use specifications to be interpreted by individual engineers. The protocol directly communicate the social conventions and expectations an agent has for the dialogue. Agents with the ability to learn could use the received protocol to plan ahead or modify its own social conventions to be able to communicate with other agents.

$$\begin{array}{l}
A :: B \xrightarrow{M_i, M_o, \mathcal{P}, O} A :: E \\
\quad if \quad B \xrightarrow{M_i, M_o, \mathcal{P}, O} E \\
A_1 \; or \; A_2 \xrightarrow{M_i, M_o, \mathcal{P}, O} E \\
\quad if \quad \neg closed(A_2) \wedge A_1 \xrightarrow{M_i, M_o, \mathcal{P}, O} E \\
A_1 \; or \; A_2 \xrightarrow{M_i, M_o, \mathcal{P}, O} E \\
\quad if \quad \neg closed(A_1) \wedge A_2 \xrightarrow{M_i, M_o, \mathcal{P}, O} E \\
A_1 \; then \; A_2 \xrightarrow{M_i, M_o, \mathcal{P}, O} E \; then \; A_2 \\
\quad if \quad A_1 \xrightarrow{M_i, M_o, \mathcal{P}, O} E \\
A_1 \; then \; A_2 \xrightarrow{M_i, M_o, \mathcal{P}, O} A_1 \; then \; E \\
\quad if \quad closed(A_1) \wedge A_2 \xrightarrow{M_i, M_o, \mathcal{P}, O} E \\
A_1 \; par \; A_2 \xrightarrow{M_i, M_o, \mathcal{P}, O_1 \cup O_2} E_1 \; par \; E_2 \\
\quad if \quad A_1 \xrightarrow{M_i, M_n, \mathcal{P}, O_1} E_1 \wedge A_2 \xrightarrow{M_n, M_o, \mathcal{P}, O_2} E_2 \\
C \leftarrow M \Leftarrow A \xrightarrow{M_i, M_i - \{M \Leftarrow A\}, \mathcal{P}, \emptyset} c(M \Leftarrow A) \\
\quad if \quad (M \Leftarrow A) \in M_i \wedge satisfy(C) \\
M \Rightarrow A \leftarrow C \xrightarrow{M_i, M_o, \mathcal{P}, \{M \Rightarrow A\}} c(M \Rightarrow A) \\
\quad if \quad satisfied(C) \\
null \leftarrow C \xrightarrow{M_i, M_o, \mathcal{P}, \emptyset} c(null) \\
\quad if \quad satisfied(C) \\
agent(r, id) \leftarrow C \xrightarrow{M_i, M_o, \mathcal{P}, \emptyset} a(R, I) :: B \\
\quad if \quad clause(\mathcal{P}, a(R, I) :: B) \wedge satisfied(C)
\end{array}$$

A protocol term is decided to be closed, meaning that it has been covered by the preceding interaction, as follows:

$$\begin{array}{l}
closed(c(X)) \\
closed(A \; or \; B) \leftarrow closed(A) \vee closed(B) \\
closed(A \; then \; B) \leftarrow closed(A) \wedge closed(B) \\
closed(A \; par \; B) \leftarrow closed(A) \wedge closed(B) \\
closed(X :: D) \leftarrow closed(D)
\end{array}$$

$satisfied(C)$ is true if C can be solved from the agent's current state of knowledge.
$satisfy(C)$ is true if the agent's state of knowledge can be made such that C is satisfied.
$clause(\mathcal{P}, X)$ is true if clause X appears in the dialogue framework of protocol $\mathcal{P}$, as defined in Figure 1.

Figure 2: Rules for expanding an agent clause

$$\langle C_i \xrightarrow{M_i, M_{i+1}, \mathcal{P}, O_i} C_{i+1}, \ldots, C_{n-1} \xrightarrow{M_{n-1}, M_n, \mathcal{P}, O_n} C_n \rangle$$

Figure 3: Sequence of rewrites

The protocol language is strictly concerned with the interaction level of communication. The semantics of the language does not depend on any assumptions about the agent's internal deliberative model. All requirements for the interaction are publicly specified with the protocol. Agents with different models of deliberation are able to communicate (McGinnis et al., 2003).

4 Means of Adaptation

Protocols are traditionally seen as a rigid ordering of messages and processing to enable a reliable means of communication. Agent-centric approaches have tended to avoid their use, lest agents be reduced to nothing more than remote function calls for the multi-agent system. The control over agent interactions within an electronic institutions is indeed intrusive. As described in section 2.1, the administrative agents of electronic institutions have complete control. The sequence of messages are dictated but also the roles an agent may adopt and the actions an agent must take within and outside of the context of the dialogue.

The protocol language of this paper does not follow this tradition. It is designed to bridge the gap separating the two approaches to agent interaction. The language is capable of representing the scenes and performative structure of electronic institutions, but it is not limited to electronic institution's inflexible model of agent interaction. The protocol language and the process of sending the protocol during execution provides agents with a means of adaptation.

In the electronic institution model, the protocol does not exist within the participating agents. It is retained by the institution itself, and designers must engineer agents that will strictly conform to the protocol which will be dictated by the administrative agents. Our approach delivers the protocol to the participating agents. Individual agents are given providence over the protocol they receive. This returns the power of the interaction to the participating agents. For example, the protocol received is not required to be the protocol that is returned.

The protocol, as described so far, already allows for a spectrum of adaptability. At one extreme, the protocol can be fully constrained. Protocols at this end of the spectrum would be close to the traditional protocols and electronic institutions. By rigidly defining each step of the protocol, agents could be confined to little more than remote processing. This sacrifice allows the construction of reliable and verifiable agent systems. At the other extreme, the protocols would be nothing more than the ordering of messages or even just the statement of legal messages(without any ordering) to be sent and received. Protocols designed this way would be more akin to the way agent-centric designers envisage agent communication. Agents using these protocols would be required to reason about the interaction to determine the next appropriate step in the dialogue. Though the protocol language is expressive enough for both extremes of the spectrum, the bulk of interactions are going to be somewhere in the middle. A certain amount of the dialogue will need to be constrained to ensure a useful dialogue can occur. This allows agents to express dynamic and interesting dialogues.

The protocol language is flexible enough to be adapted during run-time. Yet, protocols modified indiscriminately would return us to the problem facing the agent-centric approach. We would have a model for flexible communication, but no structure or conventions to ensure a meaningful dialogue can take place. It is necessary to constrain any adaptation in a meaningful way. By the examination of patterns and standards of an agent-centric approach, protocols can be construct to have points of flexibility. Portions in the dialogue can be adapted without losing the benefits of a protocol-based approach. The example below employs the rules for playing a dialogue game, the protocol language, and an amendment to the rewrite rules to allow a more dynamically constructed protocol.

5 Example

Figure 4 shows an example of an Information-seeking dialogue game similar to the one defined in Parsons et al. (2003). The dialogue game rules are simplified to clarify its implementation within the protocol. There are countless variations on the rules for any one type of dialogue game. This illustrates a continuing problem with agent-centric communication design. It is not a trivial requirement to ensure agents within a system are employing the same communicative model. This is the same with dialogue games. Subtle differences could break the dialogue. By the use of the protocol, agent can communicate their 'house' rules for the game. The rules for this particular game are as follows:

1. The game begins with one agent sending the message *question(p)* to another agent.
2. Upon receiving a *question(p)* message, an agent should evaluate p and if it is found to be true, the agent should reply with *assert(p)* else send an *assert(null)* which is a failure message.
3. Upon receiving an *assert(p)*, an agent should evaluate the assertion, then the agent can send an *accept(p)* or *challenge(p)* depending on whether the agent's acceptance attitude will allow.
4. Upon receiving a *challenge(p)*, an agent should send an *assert($\mathcal{S}$)*. $\mathcal{S}$ is a set of propositions in support of p.
5. For each proposition in $\mathcal{S}$, repeat steps 3 and 4.
6. The game is over when all propositions have been accepted or no further support for a proposition can be offered.

Rule one is satisfied by an agent taking up the role of the 'seeker'. This provides the agent with the legal moves necessary to play that side of the information-seeking dialogue game. The other agent will receive the *question(p)* message along with the protocol of figure 4. The agent identifies the clause which it should use. In this example, the clause playing the 'provider' role. It is necessary to use constraints to fully satisfy the second rule. Part of the rule states an agent sending an *assert(p)* depends on its knowledge base and its assertion attitude, otherwise an *assert(null)* is sent. The constraint *verify(p)* is assumed to be satisfiable by the agent. The agent is free to satisfy the constraint how it prefers. This could range from a simple function call to a complex esoteric belief logic with identity evaluation. The protocol only states what conditions must be satisfied, not how. The recursive steps are handled by the roles of *eval* (evaluate) and *def* (defend) which are similarly constrained. Finally, the termination rule for the game is written as the last line in the 'evaluate' role. No more messages are sent when the remainder of the set of propositions is empty.

$$\begin{array}{l}
agent(infoseek(P,B),A) :: \\
\ agent(provider(P,A),B)\ or \\
\ agent(seeker(P,B),A).
\end{array}$$

$$\begin{array}{l}
agent(seeker(P,B),A) :: \\
\ question(P)\ \Rightarrow\ agent(provider(P,A),B)\ then \\
\quad assert(P)\ \Leftarrow\ agent(provider(P,A),B)\ then \\
\quad\ agent(eval(P,B),A)\ or \\
\quad assert(null)\ \Leftarrow\ agent(provider(P,A),B).
\end{array}$$

$$\begin{array}{l}
agent(provider(P,A),B) :: \\
\ question(P)\ \Leftarrow\ agent(seeker(P,B),A)\ then \\
\quad (assert(P)\ \Rightarrow\ agent(seeker(P,B),A) \\
\leftarrow\ verify(P)\ then \\
\quad\ agent(def(P,A),B))\ or \\
\quad assert(null)\ \Rightarrow\ agent(seeker(P,B),A).
\end{array}$$

$$\begin{array}{l}
agent(eval([P|R],B),A) :: \\
\ accept(P)\ \Rightarrow\ agent(def([P|R],A),B) \\
\leftarrow\ accept(P)\ or \\
\left(\begin{array}{l}
challenge(P) \Rightarrow agent(def([P|R],A),B)\ then \\
\ assert(\mathcal{S})\ \Leftarrow\ agent(def([P|R],A),B)\ then \\
\ \ agent(eval(\mathcal{S},B)A)
\end{array}\right) \\
\ then \\
\quad \left(\begin{array}{l}
null \leftarrow R = []\ or \\
agent(eval(R,B),A)
\end{array}\right).
\end{array}$$

$$\begin{array}{l}
agent(def([P|R],A),B) :: \\
\ accept(P)\ \Leftarrow\ agent(eval([P|R],B),A)\ or \\
\left(\begin{array}{l}
challenge(P)\ \Leftarrow\ agent(eval([P|R],B),A)\ then \\
\ assert(\mathcal{S})\ \Rightarrow\ agent(eval([P|R],B),A) \\
\leftarrow\ justify(P,\mathcal{S})
\end{array}\right) \\
.
\end{array}$$

Figure 4: The information-seeking protocol

Similar protocols can be written to express the other atomic dialogue types. Real world dialogues rarely consist of a single dialogue game type. McBurney and Parsons (2002) formally describe several combinations of dialogue types. *Iteration* is the initiation of a dialogue game immediately following the finishing of another dialogue game of the same type. *Sequencing* is the similar to iteration except that the following dialogue game can be of any type. In *Parallelisation* of dialogue games, agents make moves in more than one dialogue game concurrently. *Embedding* of dialogue games occurs when during play of one dialogue game another game is initiated and played to its conclusion before the agents continue playing the first. The example involves two agents; a doctor and a patient. The patient is trying to find out whether the proposition 'patient is ill' is true (i.e. looking for a diagnosis). This is the perfect scenario to play an information-seeking dialogue game and to use the dialogue game protocol. Figure 5 and 6 shows the agent clauses as they are rewritten during the course of the dialogue.

$$\begin{array}{l} agent(infoseek(\text{“}patient\ is\ ill'', doctor), patient) :: \\ \quad agent(seeker(\text{“}patient\ is\ ill'', doctor), patient) \end{array} \tag{1}$$

$$\begin{array}{l} agent(infoseek(\text{“}patient\ is\ ill'', doctor), patient) :: \\ \quad question(\text{“}patient\ is\ ill'') \Rightarrow \\ \quad\quad agent(provider(\text{“}patient\ is\ ill'', patient), doctor) \end{array} \tag{2}$$

$$\begin{array}{l} agent(infoseek(\text{“}patient\ is\ ill'', doctor), patient) :: \\ \quad question(\text{“}patient\ is\ ill'') \Rightarrow \\ \quad\quad agent(provider(\text{“}patient\ is\ ill'', patient), \\ \quad\quad\quad doctor)\ then \\ \quad\quad assert(null) \Leftarrow \\ \quad\quad\quad agent(provider(\text{“}patient\ is\ ill'', patient), doctor). \end{array} \tag{3}$$

Figure 5: The agent clauses for the patient

$$\begin{array}{l} agent(infoseek(\text{“}patient\ is\ ill'', patient), doctor) :: \\ \quad agent(provider(\text{“}patient\ is\ ill'', patient), doctor) \end{array} \tag{4}$$

$$\begin{array}{l} agent(infoseek(\text{“}patient\ is\ ill'', patient), doctor) :: \\ \quad question(\text{“}patient\ is\ ill'') \Leftarrow \\ \quad\quad agent(seeker(\text{“}patient\ is\ ill'', doctor), \\ \quad\quad\quad patient)\ then \\ \quad\quad assert(null) \Rightarrow \\ \quad\quad\quad agent(seeker(\text{“}patient\ is\ ill'', doctor), patient). \end{array} \tag{5}$$

Figure 6: The agent clauses for the doctor

The patient begins the dialogue by taking the initial agent clause of *infoseek* which stands for information-seeking. This step is labelled *1*. The agent applies the rewrite rules to expand the seeker role and sends the *question* to the doctor agent, step *2*. The doctor receives the message and the protocol. The applies the rewrite rules and finds the only instantiation that is possible is the unfolding of the provider role. It applies the rewrite rules and comes to the *verify* constraint which it is unable to satisfy. It cannot determine the truth value of the proposition and is unwilling to defend the proposition. It takes the other half of the *or* operator and sends the *assert(null)*. Let us assume the doctor agent is a bit more clever. It cannot currently assert that the patient is ill. It has a knowledge-base and an inference engine that allows it to figure whether the proposition is true or not, and it needs some more information from the patient. The particular kind of information would depend on each patient consultation. If this diagnosis scenario was part of an electronic institution, the institution would have to represent in a state diagram every possible permutation of a diagnosis scenario. This is not practical, if not impossible.

Instead, the doctor agent can use the patterns of dialogue games to structure the interaction but allow adaptations to handle any run-time dialogical needs that may arise. In the example, the doctor agent needs to ask about a different proposition before it can answer the patient's original query. This is achieved by an additional rewrite rule shown in figure 7.

$$\begin{array}{l} A \xrightarrow{M_i, M_o, \mathcal{P}, O} A\ then\ B \\ \quad if \quad clause(P, B) \wedge isa(B, dialogue - type) \\ \\ isa(infoseek, dialogue - type). \end{array}$$

Figure 7: Additional rewrite rule

This allows the agent to graft the infoseek agent clause between any term in the protocol. These rewrites can be expanded further to represent other dialogue combinations as well as domain specific rewrite rules. The doctor's dialogue clause with the use of the embedding is shown in figure 8. The expansions and dialogue begin the same, but rather than just sending the *assert(null)*. The agent inserts the agent definition *agent(infoseek(”patient has a fever”),patient),doctor)*. The next instance of a information-seeking dialogue is begun. The moves of the embedded dialogue game are in bold text. In this instance the patient plays the provider role and the doctor plays the seeker. The game is finished by the patient asserting ”patient has a fever”. The doctor, now knowing this proposition to be true, has enough knowledge to assert the original proposition posed by the patient's first question. The first information-seeking game also concludes successfully by the doctor making the diagnosis and asserting the proposition ”patient is ill” is true.

$$\begin{aligned}
&agent(infoseek(\text{“}patient\ is\ ill''), patient), doctor) ::\\
&\ question(\text{“}patient\ is\ ill'') \Leftarrow\\
&\quad agent(seeker(\text{“}patient\ is\ ill'', doctor), patient)\ then\\
&\quad agent(infoseek(\text{“}patient\ has\ a\ fever'', patient)\\
&\qquad\qquad\qquad\qquad\qquad\qquad , doctor).
\end{aligned}\tag{6}$$

$$\begin{aligned}
&agent(infoseek(\text{“}patient\ is\ ill'', patient), doctor) ::\\
&\ question(\text{“}patient\ is\ ill'') \Leftarrow\\
&\quad agent(seeker(\text{“}patient\ is\ ill'', doctor), patient)\ then\\
&\quad \mathbf{question(\text{“}patient\ has\ a\ fever'')} \Rightarrow\\
&\quad\ \mathbf{agent(provider(\text{“}patient\ has\ a\ fever'', doctor)}\\
&\qquad\qquad\qquad\qquad\qquad\qquad \mathbf{, patient)}
\end{aligned}\tag{7}$$

...

$$\begin{aligned}
&agent(infoseek(\text{“}patient\ is\ ill'', patient), doctor) ::\\
&\ question(\text{“}patient\ is\ ill'') \Leftarrow\\
&\quad agent(seeker(\text{“}patient\ is\ ill'', doctor), patient)\ then\\
&\quad \mathbf{question(\text{“}patient\ has\ a\ fever'')} \Rightarrow\\
&\quad\ \mathbf{agent(provider(\text{“}patient\ has\ a\ fever'', doctor)}\\
&\qquad\qquad\qquad\qquad\qquad\qquad \mathbf{, patient)}\ then\\
&\quad\ \mathbf{assert(\text{“}patient\ has\ a\ fever'')} \Leftarrow\\
&\quad\ \ \mathbf{agent(provider(\text{“}patient\ has\ a\ fever'', doctor)}\\
&\qquad\qquad\qquad\qquad\qquad\qquad \mathbf{, patient)}\ then\\
&\quad assert(\text{“}patient\ is\ ill'') \Rightarrow\\
&\quad\ agent(seeker(\text{“}patient\ is\ ill'', doctor), patient).
\end{aligned}$$

Figure 8: Embedded information-seeking agent clause for doctor

6 Conclusions

The protocol language described in the paper is expressive enough to represent the most popular approaches to the agent communication. It is able to capture the various aspects of Electronic Institutions such as the scenes, performative structure, and normative rules. This enables agents to have structured and meaningful dialogues without relying on centralised control of the conversation. The language is also capable of facilitating agent-centric approaches to agent communication. Agents pass the protocol to their dialogical partners to communicate the social conventions for the interaction. Agents can adapt the received protocols to explore dynamic conversation spaces. The protocol language in this paper is not seen as a replacement for either model of agent communication. Instead, it synthesises the two approaches to gain the advantages of both. Protocols are used to coordinate and guide the agent's dialogue, but agents are able adapt the protocol by using an agent-centric model for communication. The use of this communicative model constrains transformation to the agent clauses in meaningful ways. The run-time delivery provides the mechanism for communicating the protocol as well as any adaptations that are made. We have begun developing a FIPA compliant agemts which uses the ACL library and the protocol language. It is hoped that the verifibility and semantic problems associated with FIPA's ACL can be mitigated by the use of the protocol language to communicate the performative's semantics during their use.

This approach does raise new issues which have not been addressed in this paper. One issue concerns restricting changes to the protocols. There are certainly dialogues where certain agents will be restricted from modifying the protocols or dialogue which require portions of the protocol to remain unchanged. This remains for future work along with development of a vocabulary of generic transformations which can be proven *a priori* or verified to retain semantic and syntactical continuity of the protocols.

The protocol language has already been shown to be useful for a number of agent purposes. A 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 protocols. 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 Walton (2004a). The protocol language has been used to implement the generic dialogue framework of McBurney and Parsons (2002) and the negotiation game described in McBurney et al. (2002).

References

Marc Estava, Juan A. Rodriguez, Carles Sierra, Pere Garcia, and Josep L. Arcos. On the formal specifications of electronic institutions. *LNAI*, pages 126–147, 2001.

Marc Esteva, David de la Cruz, and Carles Sierra. Islander: an electronic institutions editor. In *Proceeding of the first International joint conference on Automomous agents and multiagent systems*, pages 1045–1052, Bologna, Italy, 2002. ACM press.

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.

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

Robert A. Flores and R.C. Kremer. To commit or not to commit: Modelling agent conversations for action. *Computational Intelligence*, 18(2):120–173, May 2002.

Foundation for Intelligent Physical Agents. Fipa sl content language specification, 2000.

Mark Greaves, Heather Holmback, and Jeffrey Bradshaw. What is a conversation policy? In Frank Dignum and Mark Greaves, editors, *Issues in Agent Communication*, pages 118–131. Springer-Verlag: Heidelberg, Germany, 2000.

Nicolas Maudet and Brahim Chaib-draa. Commitment-based and dialogue-game based protocols: new trends in agent communication languages. *The Knowledge Engineering Review*, 17(2), 2002.

Nicolas Maudet and Fabrice Evrard. A generic framework for dialogue game implementation, 1998.

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.

Peter McBurney, Rogier van Eijk, Simon Parsons, and Leila Amgoud. A dialogue-game protocol for agent purchase negotiations. *Journal of Autonomous Agents and Multi-Agent Systems. (In press).*, 2002.

Jarred McGinnis, David Robertson, and Chris Walton. Using distributed protocols as an implementation of dialogue games. Presented EUMAS 2003, December 2003.

Simon Parsons, Peter McBurney, and Michael Wooldridge. The mechanics of some formal inter-agent dialogues. In *Workshop on Agent Communication Languages*, pages 329–348, 2003.

Philippe Pasquier, Nicolas Andrillon, and Brahim Chaib-draa. An exploration in using the cognitive coherence theory to automate agents's communicational behavior. In *Agent Communication Language and Dialogue workshop*, Melbourne, Australia, 2003. AAMAS'03.

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.

David Robertson. A lightweight coordination calculus for agent social norms. In *3rd International Conference on Autonomous Agents and Multi Agent Systems*, New York, USA, 2004a. paper in submission, available from author.

David Robertson. A lightweight method for coordination of agent oriented web services. In *Proceedings of AAAI Spring Symposium on Semantic Web Services*, California, USA, 2004b.

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

Munindar P. Singh. A social semantics for agent communication languages. In Frank Dignum and Mark Greaves, editors, *Issues in Agent Communication*, pages 31–45. Springer-Verlag: Heidelberg, Germany, 2000.

Wamberto Vasconcelos. Skeleton-based agent development for electronic institutions. In *First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002. AAMAS, ACM Press.

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

Chris D. Walton. Model Checking Multi-Agent Web Services. In *Proceedings of the 2004 AAAI Spring Symposium on Semantic Web Services (To Appear)*, Stanford, California, March 2004a.

Chris D. Walton. Multi-Agent Dialogue Protocols. In *Proceedings of the Eighth International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, Florida, January 2004b.

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

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