Achieving Common Interaction Protocols in Open Agent Environments

Shamimabi Paurobally
University of Southampton
Southampton SO17 1BJ, UK
sp@ecs.soton.ac.uk

Jim Cunningham *
Imperial College, London
London SW7 2BZ, UK
rjc@doc.ic.ac.uk

ABSTRACT

In this paper, we discuss the challenges of specifying and implementing agent interaction protocols with the same interpretation by all parties in an open system. This raises the issue of what it means for a group to follow and comply with a protocol. Resolving such questions is essential for facilitating agent interactions, interoperability, and trust between independently developed agents. To this end, we aim for clear and complete protocol specifications, executable protocols, and ways in which a group may adopt a common protocol for realistic agent interactions.

Keywords

Agent interaction, protocol specification and implementation

1. INTRODUCTION

Protocols for English, Dutch, sealed bid auctions, and contract nets are a few examples amongst the many protocols for possible use in agent interaction. There are even protocols for agreeing on a protocol. An interaction protocol defines possible sequences of messages that interacting agents must follow in order to achieve their goals. For each agent to follow the protocol, it must be common knowledge for the group of agents. Yet it is remarkably difficult to specify a protocol completely, in a way so that there can be no difference in interpretation amongst the agents. A recent thesis of the first author [7], and several joint papers have shown that many, if not the majority of, published agent interaction protocols are flawed in this respect. The trust of an agent in the outcome of an interaction depends on published protocols being well defined and complete, with known properties. This is even more critical for an open environment. The protocol must also be correctly implemented in each agent. There is a major challenge in achieving this. We need specification methods which can be understood by designers, yet be strong enough to ensure proven properties, and we need implementation methods, or ways of executing common protocols library which do not introduce new errors.

In principle, protocols that have not been predefined could emerge in flexible societies of agents which can adapt to unexpected events. However, agents with current technology are not intelligent enough to learn new non-trivial protocols. At present non-trivial agent conversations rely on predetermined interaction protocols. Even in this case, there remain several open issues in pre-arranging protocols.

This paper identifies some of the challenges for expediting agent interaction through protocols in an open system. We provide an outlook from both designer's and developer's points of view when developing interaction protocols. We believe that realistic interactions need not only adequate expressiveness, but also a consistent shared interpretation of the specification language between groups of agents. Such shared understanding raises issues about attaining and maintaining knowledge about a common protocol and attaining compliance with the common protocol. We seek to focus on these issues. The beginnings of our solutions can be found in [11], [10] and [7].

Section 2 of this paper provides an overview of current methods for specifying protocols, their drawbacks and some means to overcome them. Section 3 discusses the problem of a group following a common protocol given an unpredictable environment. Section 4 examines having executable libraries of protocols. Section 5 presents our conclusions.

2. SPECIFYING PROTOCOLS

A protocol defines the "rules of procedure" for a conversation. Even though there are a number of well-known protocols like auctioning protocols, normally a protocol has to be conveyed to each participant in order to serve as a point of reference. For example, a user foreign to English customs may not know the protocol for an English auction offhand. On the other hand, a protocol can also be customised according to the application domain. Furthermore, a well-specified protocol helps all participants to abide by the same rules and progress towards a consensus. This in turn depends on the specification methodology used. A vague or incomplete protocol, resulting from lack of expressiveness in the specification language, may allow different interpretations and breakdown in interaction. To avoid such misunderstandings, the specification methodology has to exhibit clear semantics and adequate expressiveness. This section discusses how published protocols can be flawed because of the specification method used. It also outlines our approach, based on extended statecharts and PDL, for protocol specification, and highlights remaining issues.

2.1 Current Methodologies

There are three methodologies predominantly used for specifying protocols – AUML, Petri nets and statecharts. We briefly discuss each of them. A more detailed critique can be found in [7] and [9].

AUML [1] extends UML sequence diagrams to represent asynchronous exchange of messages between agent roles along timelines. The advantages of AUML are its visual representation along timelines and reuse of UML constructs. However, expressing protocols of realistic complexity in AUML

*A member of the Agentcities.rtd project.

AAMAS '02 Melbourne, Australia
.

requires substantial efforts for developing, debugging and understanding. This can yield cluttered diagrams that are easy to misinterpret. One of the main problems is that, because both roles and timelines are used, identifying an agent requires a new timeline. If each agent sends a different message, then as many timelines are needed as the number of agents. So even with a set of n agents, if there are m different messages, where each agent must be identified and may send any of the messages, the diagram has to show n timelines and m messages on each timeline. An AUML diagram becomes hard to read when three or more timelines are shown with at least two different non-terminal messages possible at decision points. The problem escalates when broadcasting messages to the different timelines, expressing iterative actions or messages possible at any point such as timeouts or rejections. AUML protocols thus do not capture the multi-lateral nature of agent interactions in open systems.

Petri nets are another candidate for graphically modeling interaction protocols and have tools for detecting conflicts and their properties. However the notion of an agent executing an action is not explicit in the notation [2]. A different Petri net can be assigned to each agent role [13], raising questions about how the entire protocol is inferred and the reachability and consistency of shared places. It can be seen that the Petri net in [12] for the FIPA request protocol, even though showing only the initiator role, is more complex than in AUML [6]. If a single Petri net is partitioned for each role [5], this leads to a complex diagram where a partition is required for each agent identified. In each case, there is redundancy in repeating the same parts of a protocol for different agents or roles. Furthermore, alternative actions and states such as either *agree* or *reject* but not both, cannot be expressed in standard Petri nets.

Statecharts [4] are an extension of conventional finite-state machines. In their original form, statecharts do not portray the agents in the exchanged messages and triggered states. It is also difficult to show compound transitions for nested protocols and their results. There are situations where undefined states and conflicts between states may arise. Yet we find that it is easier to express protocols in statecharts than in AUML or Petri nets. The statecharts are less cluttered diagrams. Augmenting the statechart notation to represent multi-agent interactions is relatively simple and identifying the agents in a dynamic environment is easier than in the other notations. Statecharts do not suffer from a drastic rise in complexity with increase in the number of identified agents. Futhermore, states and processes are treated equally allowing reasoning about an interaction. This also allows us to refer to a point or a path of an interaction.

2.2 Extended Statecharts and PDL

A specification methodology must be expressive enough to provide clear and understandable protocols and be amenable to mechanised verification and symbolic reasoning. Nesting of protocols must be possible. We propose a combined graphical and logical approach by extending both statecharts and propositional dynamic logic (PDL [3]). Protocols in extended statecharts are easy to comprehend visually and protocols in extended PDL offer the benefits of a formal approach – more machine readable, verifiable and executable.

The syntax, semantics and application of the extended form of PDL can be found in [7], [10]. Here we provide a brief description of the additional constructs that are used in both extended statecharts and extended PDL. The statecharts are extended with graphical representation of the constructs that we add to PDL. Together they provide a unified logical and graphical specification language. We treat an agent (with an optional role) as capable of atomic actions or complex processes. Each atomic action constitutes a primitive process that may be combined into more complex ones. The processes performed by an agent trigger states of interaction which themselves can be organised into a hierarchy of parent and sub-states. Groups of agents are treated with set notation. In the following summary, let α denote a process and S a state:

Description	**Syntax**	**Example**
Agent and role	$Agent : role$	$\{ann{:}seller, ted{:}buyer\}$
Agent(s) doing α	$Agent \bullet \alpha$	$roger \bullet bid$
Parameterising state S with agent(s)	$S(Agent)$	$agreed(roger)$
Testing a state S	$S?$	$offered?;agree$
Process types	$\alpha_1 :: \alpha_2$	$E\text{-}bay :: English\text{-}auction$
Process composition	$\alpha_1; \alpha_2$	$offer\ ;\ agree$

An example of the application of extended PDL in specifying the contract net protocol can be found in [9]. Figure 1 expresses the FIPA request protocol [6] in extended statecharts. Let *In* denote an initiator role and *Par* denote a participant role. As can be seen, actions like *cancel* can be expressed from the *open* parent state and do not need additional meta-protocols as proposed by the latest version of the FIPA request protocol. A meta-protocol is essentially a higher-level sequence diagram without the details of a particular protocol.

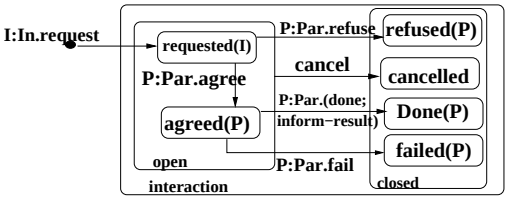

Figure 1: Request Protocol in extended statecharts.

The extended PDL theory of the request protocol is found in [9]. The graphical, in extended statecharts, and logical, in extended PDL, protocols are complementary to each other.

2.3 Issues in Protocol Specification

This section discusses open issues in protocol specification.

Providing an adequate methodology with clear semantics. Petri nets, AUML and statecharts are mostly used, but these notations are either continuously evolving or are personally adapted by each developer. As errors are found in the protocols or the methodology, the protocols and notation are patched in an ad-hoc manner. Thus, there is no widely-used methodology that has a stable specification, with clear semantics and most importantly enough expressiveness for multi-agent interactions. Our approach is to provide clearly specified and verifiable protocols and libraries for executable forms of these protocols. In an open environment, instead of the protocols being coded in the agents and then establishing similarities between personal protocols, the participants can choose protocols from libraries according to their needs, before downloading a relevant implementation of the protocol.

Incompleteness of protocols. Many protocols suffer from incompleteness for various reasons including the following:

- The specification of a protocol in a particular methodology does not meet its natural language specification. For example, as opposed to the given informal description of the protocol, the English auction protocol in AUML [6] does not model (1) storing the last highest bidder, (2) termination if no buyers send a proposal and the auctioneer no longer sends revised call for pro-

posals and (3) termination if the auctioneer rejects all proposals.

- A state may be undefined. Some agents may interpret it as true, while other agents interpret the unknown state as false. So the agents may take contradictory actions. For example, an incomplete protocol may allow a state like *agreed* to be undefined after a sequence of actions. Some agents may believe an agreement has been reached and send payment for a service, while others believe the state is not *agreed* and not deliver the service.
- Conditions may be undefined or not reset for recursive actions. Conditions for sending a message may depend on previous states, need to be initialised, set and reset, otherwise for some agents the conditions may hold, but not for others.
- The protocol actions may be vague or have only partial information e.g. when using tokens such as a_x. This should be covered by the semantics of the protocol.
- The protocol does not specify how to deal with error messages. How do agents deal with lost, delayed, unordered or corrupted messages? Interaction protocols usually do not address these issues. Error recovery protocols are another issue. How do interacting agents recover from errors in message exchange, timeouts and crashes? How is the interaction saved and restored and the consistency between the agents' mental states maintained?
- The protocol does not specify the case when an agent receives unexpected messages such as wrong or ghost messages.
- Termination may not be obvious, or a protocol may not terminate.

Libraries of protocols. Given acceptable specification methods, there can be libraries of protocols from which developers or more flexible agents may choose depending on their needs. These libraries can in turn be organised into classes of protocols located in a market-place or a repository.

Verifiable protocols. There is also no accepted notation for expressing formal protocols which can be verified before being released in the public domain. Verifiable protocols allow providers of protocols to also endorse their protocols with proofs of properties. We presume that fewer risks are involved in an interaction with a well-defined and verified protocol and so users would be more willing to cooperate and trust the multi-agent system.

To help choose between protocols. An increase in specified protocols may lead to large libraries, from which it is hard to choose a protocol. To help choose between protocols, their properties could be published, such as liveness, completeness, stability, complexity, fairness, Pareto optimality, etc. The frequency with which a protocol has been used and ratings from other users may also help.

Properties of protocols. The proof that a protocol exhibit certain properties should ideally be made available to users, as a guarantee. Security properties and those that discourage malicious behaviour are important. The extended PDL can be used to specify axioms for the properties of a protocol. It can also be used as a program logic to show how a protocol expressed in extended PDL satisfies a safety or liveness property [8]. Complexity and scalability properties of protocols need to be published to avoid the problem of explosion in the state space. Common multi-lateral protocols like English auctions exist because they are efficient and scalable. An open issue remains about whether a user can trust a claim that a protocol satisfies a desirable property, even if shown the proof. Does a user have enough expertise to check a proof?

Other issues arise in an environment where agents are trying to interact with other unknown agents and where the protocol may be the only reliable information. If agents may act irrationally or fail to comply to the protocol, what should be done?

3. COMMON PROTOCOLS

A common protocol ensures that all participants following it will coordinate meaningfully and can expect certain responses from others. But what does it mean for a protocol to be common and how does it become so? A common protocol for a group would ideally mean that all agents know the protocol, and know that others know the protocols, and know that others know they know the protocol and so forth. Such common knowledge about the protocol can be hard to attain. On the other hand, compliance with the protocol must also be commonly known. Ideally, all agents comply with the protocol, know that everyone complies with the protocol and know that everyone knows everyone complies and so forth. This seems to be even more difficult to ascertain.

There are a number of ways to gain knowledge about a protocol, for example the protocol may be dynamically prearranged upon entering an interaction. The protocol may be coded in the agents, the agents may obtain a protocol from a repository of published protocols or from a library of protocols as mentioned in the above section or an institution may dictate the protocol. Two agents prefering different protocols may agree on a sub-protocol which is a common subset of each protocol. However there are also issues about gaining such a level of knowledge.

3.1 Common Knowledge and Beliefs

In open environments, common knowledge about a protocol would be hard to achieve and require that agents spend significant resources on exchanging messages just to set the grounds for attaining a common protocol. Even more effort would be needed for explicitly determining compliance to a common protocol. A solution would be to start the interaction when everyone knows the protocol and assumes compliance, with an increasing level of knowledge attained as the interaction progresses. For example, in an open market, an agent may decide to sell a service and to broadcast the future sale around. It may then choose a protocol or accept a suggested protocol and broadcast the protocol. A number of other participants then announces that they are joining the group, and that they know and will comply with the protocol. From these premises, the open group of agents can start to negotiate. Those taking active part in the process are those who know the protocol and follow it. Gradually, it becomes known which agents are taking part in the negotiation. In this way, cooperation and organisation for executing a negotiation emerges.

One important issue remains: achieving and maintaining consistency between the agents' beliefs about the state of the interaction. An interaction state at some instance is entailed from the propositions believed by the group about that interaction. For example, an agent sends an offer to the rest of the group and a receiving agent then believes the interaction state to be offered, from which it can reply with an agreement.

3.2 The Interaction State

What happens when messages get lost or delayed over the network? It is then hard to achieve a global interaction

state. An agent may believe the state to be agreed, while another believes it to have timed out. The differences between the beliefs in the group of agents lead to inconsistency between beliefs. Such differences in beliefs are acceptable while messages are being relayed, but eventually the group has to share the same beliefs about the interaction state before progressing to the next state. That is, before progressing to an agreement state, the group must share the belief that an offered state has been triggered.

To address this issue, we can have synchronisation protocols which lie underneath interaction protocols. While interaction protocols define possible sequences of high-level communicative acts, synchronisation protocols specify how the messages are exchanged at a lower-level and how the group revise their beliefs about the interaction state. For example, one such synchronisation protocol in [11] specifies that an agent can repeatedly send a message until receiving an acknowledgement and only a receiver, and not the sender, updates its beliefs. In [11] we specify such protocols and prove the safety they achieve in the progress and termination of a bilateral interaction. More discussion and examples of synchronisation protocols can be found in [11]. Protocols for multi-lateral interactions are harder and raise lots of issues.

4. IMPLEMENTING PROTOCOLS

In open environments, agents with protocols hardcoded into them are not flexible and it may not be feasible to encode a large number of protocols into an agent. Knowing the specification of a protocol does not necessarily mean correctly implementing it. There should be methods for ensuring and proving the sound implementation of a protocol.

4.1 Executable Protocols

The problem of each user ensuring that a protocol is correctly implemented may be circumvented by placing the responsibility on the provider of a protocol. By this we mean that libraries of specified protocols could be accompanied with libraries of executable protocols. Agents could arrive in a market and download executable protocols from those libraries. The properties such as soundness and completeness of both specified and implemented protocols could be ensured by providers, who could probably have tools for doing so automatically.

Even with different agents downloading the same executable protocol, there is no guarantee that they share the same interpretation of the code. Thus we come to the idea that agents may run on a common framework that have libraries of protocols and mechanisms for executing them. The extended form of PDL for developing protocols may be a candidate for turning interaction protocols into an executable form.

4.2 Open Issues

There are a number of challenges to providing agents with executable forms of a protocol. Assuming there may be a provider of executable protocols, we mention some of these below:

- How does an agent ensure a sound implementation of a protocol by a provider?
- How does an agent choose between different executable forms of similar specifications from different providers?
- Does an agent trust a provider enough to execute a protocol in its personal environment?
- Can privacy of information of different users be protected in an open environment?
- How does a framework provider create the executable forms?
- Can an agent avoid unexpected relationships or side-effects between protocols when executing them in parallel?
- How is concurrency handled in real-time?
- Are error conditions covered by the protocol? For example, when there are timeouts, network failures or an agent has crashed.

Given these features, the implementation of protocols may benefit from research on mobile code and mobile agents.

5. CONCLUSIONS

Pre-determined interaction protocols may not seem a flexible approach given open agent systems, but complex forms of interactions and reasoning may emerge from such types of interaction. This paper has discussed some issues to be addressed for facilitating agent interaction through protocols. We need well-defined and complete protocols and notations/methodologies with enough expressiveness for the multi-agent domain, along with clear semantics and illustration. Implementations of protocol should also be verifiable and their properties validated. To this end, we have provided an overview of currently used methodologies for specifying protocols and discussed the challenges in having a group of agents know and comply with a common protocol.

6. REFERENCES

[1] B. Bauer, J. P. Muller, and J. Odell. Agent UML: A Formalism for Specifying Multiagent Software Systems. In *AOSE*, pages 91–104, 2000.

[2] R. Cost, Y. Chen, T. Finin, Y. Labrou, and Y. Peng. Modeling agent conversations with colored petri nets. In *Workshop on Specifying and Implementing Conversation Policies*, pages 59–66, 1999.

[3] R. Goldblatt. *Logics of Time and Computation*. CSLI, 1987.

[4] D. Harel and M. Politi. *Modeling reactive systems with statecharts*. McGraw-Hill, 1998.

[5] M. Nowostawski, M. Purvis, and S. Cranefield. A layered approach for modelling agent conversations. In *2nd Work. on Infrastructure for Agents, MAS, and Scalable MAS*, Agents 2001.

[6] J. Odell, H. Parunak, and B. Bauer. Representing Agent Interaction Protocols in UML. In *AOSE*, pages 121–140, 2001.

[7] S. Paurobally. *Rational Agents and the Processes and States of Negotiation*. Imperial College, Ph.D. Thesis, 2002.

[8] S. Paurobally and J. Cunningham. Safety and liveness of negotiation protocols. In *AISB2002 Intelligent Agents in virtual market track.*, 2002.

[9] S. Paurobally and R. Cunningham. Verification of protocols for negotiation between agents. In *ECAI-15*, pages 43–48, 2002.

[10] S. Paurobally, R. Cunningham, and N. R. Jennings. Developing agent interaction protocols using graphical and logical methodologies. In *Workshop on Programming MAS, AAMAS*, 2003.

[11] S. Paurobally, R. Cunningham, and N. R. Jennings. Ensuring consistency in joint beliefs of interacting agents. In *2nd Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, 2003.

[12] M. K. Purvis, S. Cranefield, M. Nowostawski, and M. A. Purvis. Mas interaction protocols in a dynamically changing environment. In *Work. Toward Application Science: MAS Problem Spaces and their implementation to achieve globally coherent behaviour. AAMAS*, 2002.

[13] M. K. Purvis, S. Cranefield, M. Nowostawski, R. Ward, D. Carter, and M. A. Oliveira. Agentcities interaction using the opal platform. In *Work. on Challenges in Open Agent Systems, AAMAS*, 2002.