A Software Tool for the Development of MAS Communication Protocols based on Conversations

Madieyna Lamine Fall
Département de mathématiques et d'informatique
Université du Québec à Trois-Rivières
Québec, Canada, G9A 5H7

madieyna@hotmail.com

Sylvain Delisle
Département de mathématiques et d'informatique
Université du Québec à Trois-Rivières
Québec, Canada, G9A 5H7
(1) 819-376-5011, ext. 3832

Sylvain_Delisle@uqtr.ca

ABSTRACT

The work we present here is mainly concerned with interagent communication, MAS communication protocols and, in particular, software tools and environments to define, experiment and evaluate actual protocols in MAS. We present a new MAS tool/environment called ProtocolBuilder, implemented in Java, which enables us to effectively support the development and experimentation of MAS communication protocols based on conversations (i.e. exchange of messages between agents).

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents and Multiagent systems.

General Terms

Design, Experimentation, Languages, Verification.

Keywords

MAS, development environments, software tools, communication protocols, conversations.

1. INTRODUCTION

We present a software tool for the development and experimentation of interagent communication protocols in MAS that not only makes it possible to create communication protocols, but also to verify the compliance of actual conversations with the protocols used in specific interagent conversations. The creation of protocols is done in a flexible and generic way, which helps reduce the difficulties involved in interagent communication development via the design of an environment (software) supporting the creation of communication protocols and verification of interagent conversations. For the MAS developer, this software environment facilitates a better comprehension of MAS communication protocols by hiding (when needed) the unnecessary details of messages and protocols, while being based on principles supporting the analysis and testing of conversation-based protocols.

2. RELATED WORK

The two MAS communication languages that can be considered as *de facto* standards are KQML and FIPA-ACL. However, agents developed in various environments using either KQML or FIPA-ACL cannot easily communicate with each other. So to establish a "standard" of communication, we must resolve several remaining issues such as the sharing of ontologies, the limited flexibility in communications protocols, and the absence of standardization in object- and agent-oriented MAS development tools. Thus, communication languages established in various approaches often have inadequacies. First, the semantics of these languages refers directly to the internal (private) state of an agent, through mental concepts. Second, they do not make it possible in practice to take part in conversations and must be used in combination with other tools, such as finite state machines, which specify the well-formedness of message sequences.

In recent years, we observe a clear trend towards the definition of communication languages based on public concepts, such as social engagements as well as on models of conversations more flexible than protocols, such as dialogue games. Many research works lead to the evaluation of different aspects to be taken into account in the resolution of various MAS communication issues. Representative of this trend are the contributions of [6] ("commitment machines"), [5] ("architecture for argumentative planning"), [2] (Albatross), [3] ("protocol of proposal"), [1] ("sets of dialogues"), and [4] ("negotiation protocols and dialogue games"). Various models have been proposed to represent protocols, manage interagent conversations, and perform their verification/validation. Among these, the basic ones are finite state machines, Dooley graphs [8], and Petri nets [9].

3. THE ProtocolBuilder TOOL

Agents communicate by exchanging messages between themselves. Each message is composed of several attributes, the principal attribute being the speech act, which represents the purpose of the message, followed by its parameters which complete the message. To allow for standardization, each message (sequence) should have a similar approach to the one proposed in the messages of the *de facto* standard communication languages FIPA or KQML.

The *ProtocolBuilder* tool allows the MAS developer to: 1) easily create inter-agent communication protocols and provides basic tools for the exploitation of interagent conversations; and 2) verify protocols used in actual interagent conversations by checking i) the conformity of conversations compared to the

specified protocols (created through the *ProtocolBuilder* tool), ii) the order of the different messages within the conversation, and iii) the completeness of messages in conversations.

In order to test the various facets of the *ProtocolBuider* tool (i.e. modeling, execution, verification, validation), we have used the well-known *Contract Net* protocol example, often applied to electronic trading. Using the Jack programming/development tool, we carried out the simulation of the conversation between two agents that adhere to (or, are supposed to) the *Contract Net* protocol. Then, using the *ProtocolBuilder* environment, we reproduced the *Contract Net* protocol, we adapted the interagent conversations used in the Jack simulation, and we applied *ProtocolBuilder's* verification modes for different conversation scenarios. Space limits prevent us from showing further details here. However, all details are available in [7].

ProtocolBuilder has the following main characteristics:

- Ease of use through the specification of communications protocols via graphical interfaces.
- Verification, via graphical interfaces, of the conformity of conversations and the completeness of speech acts parameters used within conversations relative to the followed communication protocol specifications.
- The *ProtocolBuilder* environment is composed of "libraries". These libraries increase dynamically as we create protocols, speech acts, and parameters. They are very useful to create other protocols, or to check whether protocols are enforced (or violated) during actual conversations.
- The *ProtocolBuilder* environment was entirely implemented with the Java object-oriented programming language.
- *ProtocolBuilder* can be used with several tools (AgentTool, Jackal, Jive, AFMAS, Zeus, etc.) [10], in order to simulate and evaluate interagent conversations.

4. CONCLUSION AND FUTURE WORK

Our tool does not allow the management of complex conversations based on simultaneous exchanges between agents. There are several possible ways to advance our work:

- The modeling of protocols within the *ProtocolBuilder* environment was carried out through finite state machine structures. Consequently, the creation of a protocol with *ProtocolBuilder* can only be done serially (i.e. concurrent actions are not handled). We could consider the modeling of protocols with Petri nets, thus allowing for the handling of certain concurrency aspects during protocol creation. We could then consider the execution of several concurrent actions, and then verify the compliance of such complex conversations to their associated protocols.
- A limitation of the current state of the art in MAS is that we know little about the semantics of the conversations and the relations between the speech acts and the conversations of which they form a part. One should be able to capture the semantic contents of the actions in protocols.

Dialogues can take several forms: persuasion, negotiation, deliberation, search of information, etc. Moreover, in a conversation, dialogues can change forms. A way to go forward would be to consider the creation of more elaborated protocols built from several sub-protocols. In principle, this would not only make it possible to gain a greater abstraction of protocols, but also a reduction in complexity in the verification and validation of conversations based on this concept of dialectical shifts [11].

5. REFERENCES

[1] Chaib-draa, B., Labrie, M.-A., and Maudet, N. (2003), "Request for Action Reconsidered as a Dialogue Game Based on Commitments". *Communication in Multiagent Systems, Lecture Notes in Artificial Intelligence #2650*, 284-299. Springer.

[2] Colombetti, M. (2000), "A Commitment-based Approach to Agent Speech Acts and Conversations". *Proceedings of the Workshop on Agent Languages and Communication Policies, 4th International Conference on Autonomous Agents*, 21-29, Barcelona.

[3] Flores. R.A. and Kremer. R.C. (2004), "A Principled Modular Approach to Construct Flexible Conversation Protocols". *17th Canadian AI Conference*, London, Canada.

[4] Dastani, M., Hulstijn, J., and der Torre, L. V. (2001), "Negotiation Protocols and Dialogue Games". *Proceedings of the Fifth International Conference on Autonomous Agents*, 180-181, Montreal, Canada. ACM Press.

[5] Reed, C., Long, D. and Fox, M. (1996), "An Architecture for Argumentative Discourse Planning", *Proceedings of The 1st International Conference on Formal and Applied Reasoning (FAPR'96)*, Lecture Notes in AI #1085. Springer.

[6] Singh, M. P. and Yolum, P. (2001), "Commitment Machines". *Proceedings of the 8th International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001)*, 245-257.

[7] Fall, M. L. (2004): "ProtocolBuilder : Outil logiciel de développement et d'expérimentation de protocoles de communication pour les systèmes multiagents", mémoire de maîtrise, Département de mathématiques et d'informatique, UQTR. (available via www.uqtr.ca/~delisle)

[8] Parunak. H. V. D. (1996), "Visualizing Agent Conversations: Using Enhanced Dooley Graphs for Agent Design and Analysis". *Proc. 2nd Intl. Conference on Multiagent Systems*, 275-282.

[9] Finin T., Cost R. S., Chen Y., Labrou Y., and Peng Y. (2000), "Using Colored Petri Nets for Conversation Modeling". *Issues in Agent Communication, Lecture Notes in Artificial Intelligence #1916*, 178–192. Springer.

[10] www.multiagent.com/Software/Tools_for_building_MASs/

[11] Walton, D.N. (1992), "Types of Dialogue, Dialectical Shifts and Fallacies". In F.H. van Eemeren, R. Grootendorst, J.A. Blair and Ch.A. Willard (eds. 1992) *Argumentation illuminated.* Amsterdam: SICSAT/ISSA, 133-147.

6. ACKNOWLEDGMENTS

We thank the National Sciences and Engineering Research Council of Canada (NSERC) for its financial support.