Law-Governed Linda as a Semantics for Agent Dialogue Protocols

Sylvie Doutre, Peter McBurney, Michael Wooldridge
Department of Computer Science
University of Liverpool
Liverpool L69 3BX
United Kingdom
{sd,p.j.mcburney,mjw}@csc.liv.ac.uk

ABSTRACT

Tuple spaces and the associated Linda language are a popular model for distributed computation, and Law-Governed Linda (LGL) is a variant allowing processes to have differential and secure access to tuple spaces. We propose a form of LGL as a means of implementing a multi-agent dialogue game protocol, such that utterances under the dialogue protocol are interpreted as actions on particular tuple spaces subject to certain laws. In this way, the tuple spaces, their associated law and the actions on them may be viewed as a semantics for the dialogue protocol syntax.

Categories and Subject Descriptors

F.1.1 [**Computation by abstract devices**]: Models of computation—*Relations between models*

General Terms

Languages

Keywords

Dialogue, semantics

1. INTRODUCTION

Agent researchers and developers have devoted considerable attention recently to the design and study of protocols for agent communication using dialogue games taken from philosophy, e.g., [1, 6]. Much of this attention has focused on the syntax of protocols, with perhaps less attention paid to their semantics. There are several different functions that a semantics for an agent communications language or dialogue protocol may be required to serve: (i) to provide a shared understanding to participants in a communicative interaction of the meaning of individual utterances, of sequences of utterances, and of dialogues; (ii) to provide a shared understanding to designers of agent protocols and to the designers (who may be different) of agents using those protocols of the meaning of individual utterances, of sequences of utterances, and of dialogues; (iii) to provide a means by which the properties of languages and protocols may be studied formally and with rigor, either alone or in comparison with other languages or protocols; (iv) to provide a means by which languages and protocols may be readily implemented.

Our focus here is on semantics for agent protocols which meet this last objective. Accordingly, we propose a semantics for an agent interaction protocol which translates utterances under the protocol into commands in a version of the *Linda* distributed computing language, and into actions on associated tuple spaces. Because of their adoption by SUN (under the name *Javaspaces*)[1] and by IBM (under the name *T-Spaces*)[2], tuple spaces are a popular technology for implementation of distributed computing applications.

The idea of an agent protocol implemented using *Linda* has already been proposed in [7, 4], but the originality of our work stands in the use of *Law-Governed Linda* (LGL), a variant of Linda that allows a differential and secure access to tuple spaces. This safety concerns is of great importance in the implementation of agent dialogue protocols.

In section 2, we present the tuple space theory and LGL, before indicating in section 3 how LGL can be used as a semantics for dialogue systems.

2. TUPLE SPACES AND LGL

The theory of *tuple spaces* [3, 2] was proposed as a model of communication between distributed computational entities. *Linda* is the associated programming language. The essential idea is that computational agents connected together may create named object stores, called *tuples*, which persist, even beyond the lifetimes of their creators, until explicitly deleted. In their Javaspaces manifestation, tuples may contain data, data structures, programs, objects or devices. They are stored in tuple-spaces, which are blackboard-like shared data stores, and are normally accessed by other agents by associative pattern matching. The use of shared stores means that communication between multiple agents can be spatially and temporally decoupled. There are three basic operations on tuple spaces: `out`, with which an agent cre-

[1] See: `http://java.sun.com/developer/products/jini/`.
[2] See: `http://www.alphaworks.ibm.com/tech/tspaces/`.

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

ates a tuple with the specified contents and name in a shared space accessible to all agents in the system; `read`, with which an agent makes a copy of the contents of the specified tuple from the shared space to some private store; and `in`, with which an agent makes a copy of the contents of the specified tuple from the shared space to some private store, and then deletes it from the shared space.

Tuple spaces are public-write, public-read spaces: any entity in the system may create a new tuple, and any entity may delete an existing one. A refinement of Linda, *Law-Governed Linda (LGL)* [5], established an administrative layer which authorizes all attempts to execute `out`, `in` and `read` commands according to pre-defined security and privacy policies.

More precisely, an LGL model is a 5-tuple $\langle \mathcal{C}, \mathcal{P}, \mathcal{CS}, \mathcal{L}, \mathcal{E} \rangle$ where: $\mathcal{C}$ is a tuple space; $\mathcal{P}$ is a set of *processes* that interact with each other via $\mathcal{C}$; $\mathcal{CS}$ is a set of *control states*, one being associated with each process, each one being a bag of *attributes*; $\mathcal{L}$ is the *global law* of the system, which governs the interactions of the various processes in $\mathcal{P}$ with the tuple space $\mathcal{C}$, and thus, with each other; $\mathcal{E}$ is a mechanism that enforces the law.

$\mathcal{L}$ prescribes the effects that the events which occur at the boundary between a process (called the *home process* of the event) and the tuple space should have. These effects have the form of a sequence of operations which have to be carried out in response to the event. The effects may concern the tuple space or the control state of a process. An example of an event is an `out(t)` operation invoked by a process `p`: a first effect may be to actually store the tuple `t` in the tuple space, and a second one may be to update the control state of `p`.

The law is global, in the sense that all processes have to obey the same law. However, the application of the law for a given event might depend on the control state of the process in which the event happened. The law is enforced by means of the distributed enforcement mechanism $\mathcal{E}$, which in fact consists in associating a *controller* with every process in the system. This controller deals with the application of the law according to the control state of the process, and maintains this control state. For instance, an `out(t)` operation might be possible only if the control state contains a special attribute which authorizes this operation.

3. LGL AND DIALOGUE SYSTEMS

The syntax of an agent dialogue protocol is generally based on: a set of *participants*, who may have a role, and their own knowledge and mental states; a *context* in which the dialogue takes place; the definition of the *locutions* which can be uttered in a dialogue; and a *protocol*, which specifies the utterances permitted at each point in a dialogue.

LGL can be used as a semantics for such systems, by associating elements of an LGL 5-tuple $\langle \mathcal{C}, \mathcal{P}, \mathcal{CS}, \mathcal{L}, \mathcal{E} \rangle$ to the elements of the dialogue system.

Hence, the context in which the dialogue takes place is the tuple space $\mathcal{C}$. To each participant is associated a process of $\mathcal{P}$ and its controller from $\mathcal{E}$, and a control state of $\mathcal{CS}$. The control state may contain attributes related to the role, the knowledge or the mental states of the participant. To each locution is associated a tuple. An utterance of a locution then calls one or more events which may affect the tuple space and/or the control state of the home process associated to the participant making the utterance. Rules of $\mathcal{L}$ are associated with the rules of the dialogue protocol. $\mathcal{L}$ also contains rules that ensure that only the authorized participants can insert, read or delete tuples. These rules are enforced by the controller associated with each process.

4. CONCLUSION

We have presented a novel semantics for agent communications protocols in which utterances under a protocol are translated into commands in *Law-Governed Linda* and, through them, into actions on certain associated tuple spaces. The semantics could readily be applied to dialogue protocols such that of [1]. Translation of agent dialogue utterances into the commands of a programming language may be seen as merely mapping one syntax into another. However, the LGL programming language commands are understood in terms of their effects on actual shared memory stores (tuple spaces), and so this translation may be viewed as a semantic mapping — from agent utterances under a dialogue protocol to actions in a real world (online). As mentioned in the Introduction, we believe that the popularity and simplicity of tuple space models such as *Javaspaces* means that our semantics will facilitate implementation of agent dialogue protocols. The LGL semantics provides a connecting bridge between the formal specification of a protocol, and its software implementation.

5. ACKNOWLEDGMENTS

The authors are grateful for partial support from the European Commission, through the ASPIC Project (IST-FP6-002307).

6. REFERENCES

[1] L. Amgoud, N. Maudet, and S. Parsons. Modelling dialogues using argumentation. In E. Durfee, editor, *ICMAS 2000*, pages 31–38, Boston, 2000. IEEE Press.

[2] N. Carriero and D. Gelernter. Linda in context. *Communications of the ACM*, 32(4):444–458, 1989.

[3] D. Gelernter. Generative communication in Linda. *ACM Trans. Programming Lang. & Systems*, 7(1):80–112, 1985.

[4] J. P. McGinnis, D. Robertson, and C. Walton. Using distributed protocols as an implementation of dialogue games. In M. d'Inverno et al., editors, *EUMAS 2003*, Oxford, 2003.

[5] N. H. Minsky and J. Leichter. Law-governed Linda as a coordination model. In P. Ciancarini et al., editors, *Object-based Models and Languages for Concurrent Systems*, LNCS 924, pages 125–146. Springer, Berlin, Germany, 1995.

[6] H. Prakken. On dialogue systems with speech acts, arguments, and counterarguments. In M. Ojeda-Aciego et al., editors, *JELIA-2000*, LNAI 1919, pages 224–238, Berlin, 2000. Springer.

[7] W. Vasconcelos, D. Robertson, C. Sierra, M. Esteva, J. Sabater, and M. Wooldridge. Rapid Prototyping of Large Multi-Agent Systems through Logic Programming. *Annals of Mathematics and Artificial Intelligence, special issue on Logic-Based Agent Implementation*, 41(2–4):135–169, 2004.