CDT: A Tool for Agent Conversation Design

Mark T. Greaves Heather K. Holmback

Applied Research and Technology
The Boeing Company
P.O. Box 3707 MC 7L-43
Seattle, WA 98124-2207
{mark.t.greaves,heather.holmback}@boeing.com

Jeffrey M. Bradshaw

Institute for Human and Machine Cognition
The University of West Florida
Bldg. 79/Rm. 196, Pensacola, FL 32514
jeffrey.m.bradshaw@boeing.com

Abstract

The design, implementation, and verification of agent conversations (*i.e.*, multiagent sequences of language primitives which are intended to together achieve some goal) can be difficult and exacting, and often requires skills which agent developers do not possess. In this paper, we describe the architecture of a planned software tool that is designed to help developers with this task. Our proposed *Conversation Design Tool* (CDT) will be built as a set of extensions to an automatic theorem-proving framework, and so will be able to directly manipulate expressions in a variety of ACL semantics description languages. The CDT will also employ a variety of graphical representations, such as finite state machine diagrams and Petri net diagrams, to facilitate reasoning in certain specialized domains. Finally, the CDT will include a generative capability to automatically create conversation policy configuration files for KAoS-architecture agents.

Introduction and Motivation

The high-level communication structures that are used by many of today's agent-based systems are at roughly the same level of development which lower-level distributed computing protocols occupied 30 years ago. Modest groupings of semi-intelligent agents have been able to successfully communicate and work together on a variety of predetermined tasks, typically supported by custom high-level protocols, a largely *ad hoc* and intuitive understanding of the protocol semantics and pragmatics, and the focused activity of a small number of closely-collaborating development teams. However, complex issues involving predictable agent behavior in large heterogeneous environments, the theoretical foundations of agent collaboration, and generalizable control strategies have often been finessed or ignored in favor of deploying working systems. As a result of this disconnect between theory and practice, we now have a reasonable understanding of many of the systems-engineering issues in industrial agent design, but our fielded agent systems often exhibit serious problems of interoperability and predictability. Essentially, in order to author functioning agents within the constraints of an industrial context and timeline, it is very tempting for programmers to shortcut the directives of agent theories. The present authors believe that this issue must be addressed before we can easily create true industrial-strength agents that can be trusted to interoperate and collaborate in service of human goals. The tools we are developing as part of Boeing's JumpStart effort (Bradshaw *et. al.* 1998) are aimed at this need.

In particular, there are several areas in which agent theory seems to be outrunning actual agent implementations. These areas include the design of security policies for agents which function in hostile or less-than-trustworthy environments, and the design of sophisticated lightweight agent planners. Boeing is actively developing tools to aid agent developers in both of these areas (Bradshaw *et. al.* 1998). However, in this paper we will concentrate on a project which involves leveraging the ongoing developments in the semantic and pragmatic foundations of different agent communication languages (ACLs). By trying to define the precise meanings of different agent communicative acts, and to bind them together into naturally occurring patterns, researchers have found that they can analyze agent conversations by leveraging existing theories from linguistic semantics and speech-act theory (*e.g.* Austin 1962; Searle 1969). These theories give us a framework with which we can understand the role and purposes of agent communicative behavior, and promise principled ways of designing agents which will exhibit the same high-level interoperability and scaleable collaboration that humans do.

Merely constructing theories, however, begs the question of how these accomplishments can inform actual agent implementations in non-research environments. How can we ensure that the designers and developers of agent-based systems properly incorporate the results of this work into their systems? Full appreciation of the most recent developments in ACL theory often requires sophisticated knowledge of speech-act theory, formal semantics, linguistic pragmatics, modal logic, possible-worlds structures, and other disciplines not normally part of a developer's (or researcher's!) skill set. We expect that, as agent technology penetrates more and more into the mainstream of programming techniques, a greater fraction of actual agent development will be done by experts in the application domain, rather than experts in distributed AI. Further, it is clear that the continuing evolution of agent technology will result in the creation of agents of widely varying degrees of sophistication, running on many different types of hard-

ware, and interacting with each other on many different levels. It is unreasonable to require that all of these agents, from the simplest to the most complex, will be able to reason at the advanced levels required by the semantics of many of these ACLs. Rather, actual agent ensembles will feature diverse levels of reasoning and communication ability. Many agents will be small and simple, some will have medium-scale reasoning capabilities, and a few will exhibit sophisticated, explicit reasoning involving beliefs and goals. And, even though each of these agents may have been designed to conform to the same overall protocols and use the same low-level message transport mechanisms, the range and type of conversations in which an agent can participate will necessarily vary widely according to resource constraints and the complexity of the task which the agent performs. For the working agent developer, it is very difficult to understand the consequences of different agent communications implementation choices, and predict how agents will behave in the complex environment of modern agent ensembles.

A simple example of the type of situation a developer can face will make the foregoing discussion clearer. Suppose that an application design specifies that a group of agents will allocate some shared resource **R** by holding a periodic auction. Let us assume that the ACL used by the agents does not contain a predefined auction sequence. The agent **A** which currently owns **R** has to advertise to the other agents that an auction for **R** will take place – *i.e.*, that **A** commits to delivering **R** to the winning bidder, and that the auction will be conducted according to certain parameters (for simplicity, assume a traditional first-price open-cry auction). What type of speech act and corresponding ACL expression should the developer select for **A** to convey the appropriate meaning to the other agents? Does **A** issue an INFORM to the other agents, explicitly telling them of **A**'s commitment to transfer **R** to the winning bidder? Or, should **A** directly OFFER to deliver **R** to some other agent, conditional on that agent supplying **A** with a winning bid? Alternatively, **A** might REQUEST that other interested agents bid for **R**, and leave it to the other agents to deduce that the winning bidder will receive **R**. Or, perhaps **A** should just QUERY the other agents for their current valuations of **R**. Clearly, there are several non-equivalent options for **A**'s behavior. Choosing a message that precisely conveys the desired content to the other agents (which may not have been written by the same developer) requires that the agent developer be fluent in the semantic theory of the ACL and the formal entailments which other agents will assign to each possibility. Without this knowledge, the selection of message for **A** will be without crucial guidance, with the accompanying negative effects on agent interoperability.

Examples like this entail for us that agent developers will need sophisticated tools to help with ACL design and implementation. To this end, Boeing is developing a prototype *Conversation Design Tool* (CDT) which developers can use to analyze the conversation structures in which their individual agents are likely to engage. This tool is primarily intended to help agent developers better understand the effects that different choices in agent communications policy will have in a design, and best craft these choices to the capabilities of each individual agent. However, because the CDT will be built on a formal theorem-proving framework, we expect that it will also provide basic logical validation and verification of proposed agent conversations, relative to an underlying semantics and theory of agency. In doing this, our tool will force ACL designers and agent implementers to make explicit many of the assumptions which underlie the higher-level conversation models, which will itself be very useful. However, in the same way that the use of an IC-CAD system has become an essential part of the contemporary design and verification of digital circuits, we envision that in the future, design tools like the CDT will play an increasingly important role in developing and fielding reliable agent software. Boeing's KAoS agent framework (Bradshaw 1997) will be the initial target for the CDT, but we intend to build the tool flexibly enough so that it can be straightforwardly extended to different ACL semantic models and communications architectures. Furthermore, because we plan to use Stanford University's OpenProof logic framework as the core of the CDT, the CDT will be able to flexibly employ graphical representations of the agent's situation without sacrificing deductive rigor. With its combination of logic and graphics in a single tool, we hope that the CDT will contribute to more valid and robust designs for all agent capability levels.

Currently, the CDT is a software architecture and a vision. It is not yet implemented in code, although important pieces of it have been written over the years in the course of various other projects. Of necessity, therefore, this paper will be less concrete than we would like. However, we believe that the CDT and tools like it represent an important new direction in tools for the agent developer. In this paper, we describe our concept for the CDT.

The Theoretical Context of the CDT

As is clear from the foregoing, for us the most salient feature of agent interaction is the way in which it is dependent on and supported by a very powerful system of communication. We analyze this system of communication in terms of *conversation policies* (Bradshaw 1997). In its most general form, a conversation policy is simply a set of abstract constraints on the allowable sequencing of primitive speech acts. Many current ACLs consist of a set of protocol primitives that explicitly signify speech acts (also called *performatives)* such as "INFORM," "OFFER," and "QUERY." Speech acts are often represented one-to-one by message types in an individual ACL, and it is via this often-implicit mapping that schematic conversation policies can be linked to specific message sequences. Thus, conversation policies directly constrain sequences of abstract speech acts, and only derivatively constrain message types or ACL verb usages via the mapping from speech acts to ACL expressions or expression sequences.

The current semantic theories supplied for ACLs range from the very informal (*e.g.*, the English descriptions given in early KQML) to the very formal (*e.g.* the compositional joint intention semantics proposed in (Smith and Cohen 1996)). We agree with Smith and Cohen that it is important to have a clear and precise semantics for an ACL, but we also believe that this is not sufficient by itself to capture all the conditions on agent communicative behavior. Agent communication is more than agents simply firing independent performatives at one another. An ACL is a *tool* used by agents to achieve larger communicative goals. Agents *use* their language to negotiate the provision of resources under constraints, to dynamically form teams with joint commitments, to interact in the pursuit of goals and subgoals – that is, agents always employ speech acts in an overall discourse context. This type of agent interaction requires a language that is strong enough and flexible enough to support the higher-level abstractions of *coordinated* speech acts and dialogs, instead of simply supporting the transmission of unadorned facts via stand-alone primitives. Therefore we also need to investigate the *pragmatic* aspects of ACLs, *i.e.* how the elements of the ACL combine to form meaningful conversations and what type of overall usage conditions will govern these conversations in the context of situated agent communication.

In the past, the authors have used small finite state machines to represent the allowable speech act sequences in KAoS conversation policies. However, finite state machines are only one possible way to formally represent constraints on speech act sequences. More expressive formalisms can be constructed out of statements in a suitable dynamic logic; less expressive (but perhaps more readily understandable) formalisms could be built out of regular expression grammars. It is not our intention to limit the types of speech act constraints that a conversation policy could encode. Nevertheless, we have found that a certain subclass of conversation policies (those which can be expressed as small finite state machines) provide a useful level of abstraction when analyzing different KAoS message sequences.

One could also imagine a conversation policy consisting of a higher-level set of constraints which, rather than specifying the exact sequence and type of speech acts involved, allowed for some degree of flexibility. Such a policy, for example, might only describe a relative sequence of "landmarks" in a conversation, where each landmark defines a set of properties that must hold of the agents involved (*e.g.* belief that an offer has been made or one has been accepted). Policies of this type provide a pathway to truly *emergent* agent conversations. By this we mean that we would eventually like agents to be able to participate in dialogs that do not rigidly conform to preexisting finite state machines which were previously set out by designers (the current representation of conversation policies). Humans routinely communicate in this fashion: although our conversations often fall into patterns, we construct meaning "on the fly," and dynamically replan conversations to suit changing circumstances and communication goals. By explicitly focusing on the microstructure of conversations and the goals and plans which drive them, we hope to spur the development of agents that treat communication as a *variety of action*, and explicitly plan speech acts on par with other actions which an agent might perform. In essence, this is the result (and the promise!) of using the framework of speech-act theory in the context of ACLs.

Several agent systems include facilities for predefined conversation policies; in fact KAoS is built around agent conversations as a meaningful unit of dialog. Often, however, these types of policies are conceived only in terms of the basic sequencing of messages, and leave many important elements of the semantics and pragmatics of these conversations either entirely implicit or only briefly described. For example, although it is common for agent systems to include a method for an agent to make or accept an arbitrary offer, none formally describe the complete conditions of use under which an offer should be made or accepted or revoked, nor do they precisely address the role that an agent's offer would play within a larger dialog structure. These issues are not part of the pure syntax and semantics of an ACL but rather can be viewed as belonging to the pragmatics of agent conversations. They go beyond the formal logical frameworks of the agent's ACL, and touch basic assumptions about the models with which an agent's correct communicative behavior is specified.

So, as part of our work in developing the CDT, we will be drawing on relevant work in linguistic semantics, pragmatics, and communication theory to apply to agent conversation behavior. For example, there are properties of human conversation that may prove useful for specifying and describing agent conversations, such as some form of the Gricean maxims of conversation (Grice 1975). On the other hand, there are many aspects of human natural language conversation that are *ex hypothesi* not relevant to the type of limited agent-agent communication we envision analyzing. (These aspects include elements such as politeness constraints; rhetorical strategies such as holding the floor; the deliberate use of metaphor and ambiguity; uncooperative or untruthful agents; anaphora and other problems of reference; etc.) Our initial approach will build on the current KAoS system design, which includes conversation policies that enable agents to simply and predictably coordinate frequently recurring interactions of a routine nature. We intend to develop this existing conversation policy framework by trying to fully specify the semantic and pragmatic conditions relevant to the various KAoS speech acts and conversation policies, and to represent them within the CDT. This, by itself, will force us to be extremely precise about our assumptions. Further, in doing this, we hope to advance the current state of theories of agency and ACL semantics and pragmatics.

We project that our level of effort in the investigation of the semantics and pragmatics of agent conversation in the CDT undertaking will be a significant fraction of the total. However, to describe it further would take us beyond the scope of this paper. We turn now to a description of the architecture of the tool we plan to construct.

Technical Description of the CDT

Formally, the CDT will implement a species of heterogeneous reasoning system (HRS) which has been specialized to support reasoning using the sorts of representations and deductions which have been found to be relevant to the analysis of the semantics and pragmatics of agent dialogs. The basic concept of a heterogeneous reasoning system has been described in several places (Barwise and Etchemendy 1994, Barker-Plummer and Greaves 1994, Greaves 1997, Allwein and Barwise 1996). In brief, a HRS is a composite reasoning support system which includes multiple component logical subsystems, each with its own syntax, semantics, and proof theory, and which also includes inference rules which operate between the different subsystems. The goal of an HRS is to provide a logically rigorous environment in which a user's reasoning using multiple different representations can proceed simultaneously, and which supports the logically sound transfer of intermediate results among the component reasoning systems. An HRS is not itself an automatic theorem prover, although it may incorporate components which implement automatic reasoning over some defined logical fragment. Rather, an HRS is a formal reasoning *environment* in which a user can employ the resources of several different representation and inference systems in order to reason about a particular domain.

Stanford University is developing the OpenProof system as an extensible framework that allows a logician to build a custom HRS for an identified domain. OpenProof currently includes implementations of several simple types of logical subsystems. These include both graphically based logical subsystems (for reasoning involving, *e.g.*, Venn diagrams and blocks worlds) and sententially based logical subsystems (for reasoning using the representations of classical first-order and modal systems). Importantly, though, OpenProof also includes a sophisticated framework for linking the various component subsystems together to yield heterogeneous proof trees. Its design supports adding proof managers for different logics, and also supplies a method to define inference rules that bridge the different deductive subsystems. OpenProof is implemented in Java, and is designed around an architecture (based on Java Beans) which allows additional user-defined deductive subsystems to be smoothly integrated into the overall HRS framework.

The CDT will bind together a particular, identified set of logical subsystems that are found to be useful for reasoning about and modeling conversations in particular ACLs. Because part of our research involves identifying these useful deductive subsystems, the precise collection of components in the CDT has not yet been set. Our strategy will be initially to select a base set of logical subsystems for the CDT, and to evaluate this selection using a group of KAoS developers at Boeing. On the basis of their feedback about usability and appropriateness to their problem domain, we will expand and modify our base set of representations. We are planning to evolve the selection of logical systems represented in the CDT over the next three years, and we hope to arrive at the end of that time with a robust and useful framework for developers to reason about agent systems. In its initial incarnation, though, the CDT will provide at least the following types of deductive systems to its users:

- **A natural deduction reasoning system for standard first-order logic.** This will allow the user of the CDT to perform reasoning using classic natural deduction introduction and elimination techniques over a first-order language with equality. This subsystem will also include automatic theorem provers for this logic. OpenProof currently includes two such theorem provers: one based on resolution and one based on the matrix method, as well as a specialized variant of the matrix method prover which incorporates domain knowledge to increase its speed in a certain class of domains.
- **A Petri net deductive system.** Petri nets are a common graphical formalism for modeling systems which are driven by discrete events, and whose analysis essentially involves issues in concurrency and synchronization. They have a fairly simple and intuitive semantics based on state transition graphs, and a well-understood set of update rules. Thus, we believe that they are an important tool with which to investigate communication and coordination in agent systems. The basic CDT will contain a Petri net reasoning tool integrated as an OpenProof subsystem.
- **An Enhanced Dooley Graph (EDG) deductive system.** Enhanced Dooley graphs are a variety of finite state machine (FSM) diagram developed by Parunak (Parunak 1996) for the analysis of agent dialogs at a speech-act level. They occupy an attractive middle ground between FSMs that label nodes with participants and FSMs that label nodes with dialog states. Singh has done some recent work using EDGs as a descriptive mechanism for his ACL semantic theories (Singh 1998). We are hoping to implement an EDG subsystem and editor as an OpenProof plug-in module for the CDT.
- **A Venn deductive system.** This is an enhancement of the traditional Peirce/Venn diagrammatic reasoning system, along the lines of Shin's *Venn-II* (Shin 1994). It is a very general system for reasoning about groups of objects and their membership relations in simple set theory, and so will support simple reasoning about the properties of groups of agents.

We are aware that many of the types of deductions (*e.g.*, temporal and epistemic reasoning) required by the most sophisticated theories of agency are modeled using a quantified modal logic. These logics are notoriously difficult for people to understand and work with. Furthermore, automated deduction systems for these types of logics are in their extreme infancy, and there are technical issues (related to the complexity of reasoning in these logics and their fundamental undecidability) which entail that automatic theorem-proving in quantified modal logics will always lag the performance of first-order theorem-provers. The CDT will not, of course, make these formal issues go

away. However, we are currently researching several ways to use specialized diagrams for particular modal domains of interest (*e.g.*, timelines and temporal graphs) in order to simplify reasoning with these logics. We also hope that by artificially limiting the expressive power of these logics in a way that is sensitive to our intended reasoning domain, we will be able to make reasoning in these logics more palatable for the CDT's users.

Through the integration of these representational and deductive tools, the CDT will support reasoning about several different types of properties of ACLs. For example, one of our goals is to explore the logical consequences of the Cohen/Levesque/Smith theory of agency (Cohen and Levesque 1995) (Smith and Cohen 1996). We would like to verify that the semantic properties of the conversation policies that are contemplated for the KAoS system would result in team formation (in the sense of (Smith and Cohen 1996)). Currently, deriving these results requires familiarity with modal logic, plus a fair amount of comfort with formal proof techniques. However, by importing the same problems into the reasoning environment of the CDT, we hope to see decreases in proof complexity, coupled with increases in proof readability and usability for those not trained in logic. These sorts of results were observed in evaluations of the Hyperproof graphical logic environment (OpenProof's predecessor) when compared with more traditional methods of logic instruction (Stenning 1995). We are optimistic that these results can be extrapolated to the sorts of reasoning about ACLs which we target with the CDT. Essentially, we believe that using structured graphics, or *mediating representations*, to carry part of the cognitive load in reasoning will result in a simpler and more intuitive environment in which to explore the conversational possibilities in a given ACL. Because of this, we hope that agent designers and developers who have had only a minimum of training will be able to use the CDT to explore and verify individual conversation policies.

Changing the Process of Agent Design

Let us return to our overall view of the industrial agent implementation environment. One of the defining features of all agent architectures is their reliance on communication with other agents to achieve their ends. An agent never stands alone. Because of this, the integration and coordination of the various agents is a dominant problem in constructing full-scale agent systems. This is true even if the agents were all designed at the same time, by people in the same organization, with the acknowledged final goal of bringing them together into a single system – an idealized development environment which is rarely achieved in large-scale industrial projects. When the problem-solving environment is conceived from the start as composed of ensembles of independently developed distributed agents, which can be dynamically brought together on a task-specific basis, the coordination problem stands squarely front-and-center. How can we ensure that the different agents that work together on a particular task, each with their own reasoning and communicative abilities, can reliably coordinate with each other via a shared ACL?

We see the main contribution of the CDT as a vision for changing the *process* of agent design and development in this area. Recall our contention that the actual construction of most agents will be carried out by *domain experts* – people who can be expected to know a great deal about the intended role of the agent and the type of information with which it will deal, but who will probably not know very much about how to formally structure the agent's communicative behavior. In much the same way that parser generators like **yacc** made it possible for everyday programmers to incorporate sophisticated formal language parsing capabilities in their applications, we hope that the CDT will enable agent developers who are not ACL experts to design agents with a high degree of ACL competency. Although the CDT will initially be aimed at the leveraging the existing KAoS-based agent capabilities at Boeing, there are several natural avenues for extension:

- Our design of the CDT ensures that the internal logics and semantic assumptions are modular and replaceable, just as the external representation editors are. This means that it will be natural to extend the CDT to other types of dialog semantics and theories of agency. All of the architectural components of the CDT will be implemented in Java.
- We also envision that the CDT will include a simple code generation capability (initially targeting KAoS-architecture agents, but extendable to other architectures). After all, the result of a developer's interaction with the CDT is a precise specification of an agent's communication policies and behavior. Instead of forcing the developer to independently author code which implements the communication policies she has arrived at, it would be natural to include in the CDT the capability to automatically generate pieces of code which would implement these policies at a state machine level.
- We hope to incorporate an initial capability to specify some aspects of emergent agent conversation. Our approach will build on the representations that we will have in place for specifying conversation policies and their conditions of use and allow for more emergent behavior at certain junctures in the conversation policy. Initially this might involve something like unspecified iterations of clarifying a request or offer/counteroffer sequences.

Conclusion

It is a commonplace of computer science that, "a really powerful tool can change its user." That, in a nutshell, is what we hope to achieve with the CDT. As computer scientists, we are here functioning in our familiar role: we create the tools and underlying agent theory, which in turn leverage and extend the capabilities of the domain experts who will develop the individual agents. Our architecture for the CDT also draws from several disciplines besides computer science: linguistics and philosophy (for speech-act

theory and pragmatics); formal semantics (for the technical analysis of meaning and commitment); advanced AI (for dialog planning), and computational logic (for verification and multirepresentational deduction). Once agent developers are given the ability to describe and verify the various syntactic, semantic, and pragmatic properties of their communications frameworks, using a combination of graphical and sentential representations, we expect that the agent systems they create will become more reliable and predictable. This directly supports our overall goal of supporting the development and deployment of reliable industrial agent ensembles.

Acknowledgements. The work described in this paper is supported in part by a contract from DARPA's Agent Based Systems Program to The Boeing Company, the University of West Florida, IntelliTek, and Sun Microsystems; by grant R01 HS09407 from the Agency for Health Care Policy Research to the Fred Hutchinson Cancer Research Center; and by the Aviation Extranet joint-sponsored research agreement between NASA Ames Research Center, The Boeing Company, and the University of West Florida. OpenProof is the result of a joint research project between the Logic Software Group at Stanford's Center for the Study of Language and Information, and Indiana University's Visual Inference Laboratory.

References

Allwein, G., and Barwise, J. (eds.). 1996. *Logical Reasoning with Diagrams.* New York: Oxford University Press.

Austin, J. L. 1962. *How to Do Things With Words.* Oxford: Oxford University Press.

Barker-Plummer, D., and Greaves, M. 1994. Architectures for Heterogeneous Reasoning: On Interlinguae. In *Proceedings of the First Conference on Inference in Multimedia and Multimodal Interfaces.*

Barwise, J., and Etchemendy, J. 1994. *Hyperproof.* Stanford, CA: CSLI Publications.

Bradshaw, J. M. 1997. KAoS: Toward An Industrial-Strength Open Agent Architecture. In J. M. Bradshaw, ed., *Software Agents.* Cambridge, MA: AAAI Press/MIT Press.

Bradshaw, J. M., Greaves, M., Holmback, H., *et. al.* 1998. Mediating Representations for an Agent Design Toolkit. In *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling, and Management.*

Cohen, P. R., and Levesque, H. 1995. Communicative Actions for Artificial Agents. In *Proceedings of the First International Conference on Multi-Agent Systems.*

Greaves, M. 1997. *The Philosophical Status of Diagrams*. Ph.D. diss., Dept. of Philosophy, Stanford University.

Grice, H. P. 1975. Logic and Conversation. In Cole, P. and Morgan, J., eds., *Syntax and Semantics: Speech Acts.* New York: Academic Press.

H. Van Dyke Parunak. (1996). Visualizing Agent Conversations: Using Enhanced Dooley Graphs for Agent Design and Analysis. In *Proceedings of the Second International Conference on Multi-Agent Systems*.

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

Shin, Sun-Joo. 1994. *The Logical Status of Diagrams*. New York: Cambridge University Press.

Singh, Munindar P. 1998. Developing Formal Specifications to Coordinate Heterogeneous Autonomous Agents. Unpublished.

Smith, I., and Cohen, P. R. 1996. Toward a Semantics for an Agent Communication Language Based on Speech Acts. In *Proceedings of the AAAI National Conference on Artificial Intelligence*.

Stenning, K., Cox, R., and Oberlander, J. 1995. Contrasting the Cognitive Effects of Graphical and Sentential Logic Teaching: Reasoning, Representation, and Individual Differences. *Language and Cognitive Processes*, 10(3/4):333-354.