

What Is a Conversation Policy?

Mark Greaves, Heather Holmback, Jeffrey Bradshaw

Mathematics and Computing Technology
The Boeing Company
P.O. Box 3707 MS 7L-43
Seattle, WA 98124-2207
{[mark.t.greaves](mailto:mark.t.greaves@boeing.com), [heather.holmback](mailto:heather.holmback@boeing.com),
[jeffrey.m.bradshaw](mailto:jeffrey.m.bradshaw@boeing.com)}@boeing.com

Abstract. In this paper we define the concept of *conversation policies*: declarative specifications that govern communications between software agents using an agent communication language. We discuss the role that conversation policies play in agent communication, and suggest several subtypes of conversation policy. Our reasoning suggests, contrary to current transition net approaches to specifying conversation policies that conversation policies are best modeled as sets of fine-grained constraints on ACL usage. These constraints then define the computational process models that are implemented in agents.

1. The Roots of Conversation Policies

The dream of agent interoperability is commonly thought to rest on three main characteristics shared by the interoperating agents:

1. They would be able to access a set of shared infrastructure services for registration, reliable message delivery, agent naming, and so forth (*i.e.*, there must be *structural* interoperability);
2. They would share (possibly through translation) a common content ontology, truth theory, and method of binding objects to variables (*i.e.*, there must be *logical* interoperability); and
3. They would agree on the syntax and semantics of a common agent communication language (ACL) in which to express themselves (*i.e.*, there must be *language* interoperability).

In the last few years, international standards bodies (*e.g.*, OMG, FIPA) and government-sponsored research efforts (ESPRIT, DARPA CoABS) have attempted to address these three aspects of agent interoperability. A surprising thing that all of this work has shown is the incompleteness of this list of interoperability characteristics: it is not difficult to construct a group of agents which satisfies all of them, and yet which cannot usefully interoperate. One common problem occurs because the above characterization of language interoperability is not broad enough. Specifically, for logically powerful and expressive ACLs like KQML [6] and the FIPA ACL [14], language interoperability requires more than simply that the agents agree on the for-

mat and meaning of the various primitive ACL messages. As a practical matter, agents must also agree on the range of possible sequences and contents of messages when they are interpreted in the context of larger goal-directed interagent dialogues, or *conversations*.

Why aren't existing ACL specification techniques sufficient for this task? Current methods for defining ACLs are built around complex and technically arcane methods for making precise the syntax and semantics of each message type. In the most advanced of these, the semantics are also formally compositional, in that there are well-defined ways to derive the meaning of a sequence of two or more messages from the meanings of the constituents [5]. However, compositional semantic theories for ACLs often do not uniquely specify the actual content and sequencing of agent messages needed to achieve to a given communicative goal. This gives rise to a significant ambiguity problem for agents that need to interact using a powerful ACL, which we will call the *Basic Problem*:

Modern ACLs, especially those based on logic, are frequently powerful enough to encompass several different semantically coherent ways to achieve the same communicative goal, and inversely, also powerful enough to achieve several different communicative goals with the same ACL message.

Put another way, the Basic Problem states that for powerful ACLs, there is a many-to-many mapping between the externally visible messages an agent produces and the possible internal states of the agent that would result in the production of the message. This would be a significant but manageable problem, except that agent interaction does not consist of agents lobbing isolated and context-free directives to one another in the dark. Rather, the fact that problems of high communicational complexity may be delegated to agents dictates that those agents must participate in extended interactions. Because agents are autonomous, they will need to independently optimize their own utility functions in these interactions, and hence they must use their beliefs about the probable goals of the other participants in order to determine the best next message to generate. And, because knowledge of the semantics of ACL messages provides only an imperfect guide to these goals (due to the Basic Problem), it is nearly impossible for an agent to reliably infer the intentions and goals underlying another agent's use of a particular ACL message.

The Basic Problem is not limited to agent communication; indeed, it is a prime characteristic of human communication in natural language. There are an infinite number of ways to express a given meaning and communicative intent in a natural language such as English, and the same English utterance can be used to achieve an infinite, or at least large, number of communicative functions. In fact, a large part of speech act theory is concerned with explaining how a certain communicative intention can be expressed by a certain utterance in a certain context, given the expressive power of natural languages. The Basic Problem is not a prohibitive one in human communication because the use of language is always situated in some context and humans are in general well equipped to use that context to interpret linguistic utterances.¹ Communication breakdowns in natural language conversations are largely due to the lack of sufficient shared context in which to interpret an utterance, not the

¹ In fact, the compactness of expression that the Basic Problem entails is one of the features that allows natural languages to be learnable.

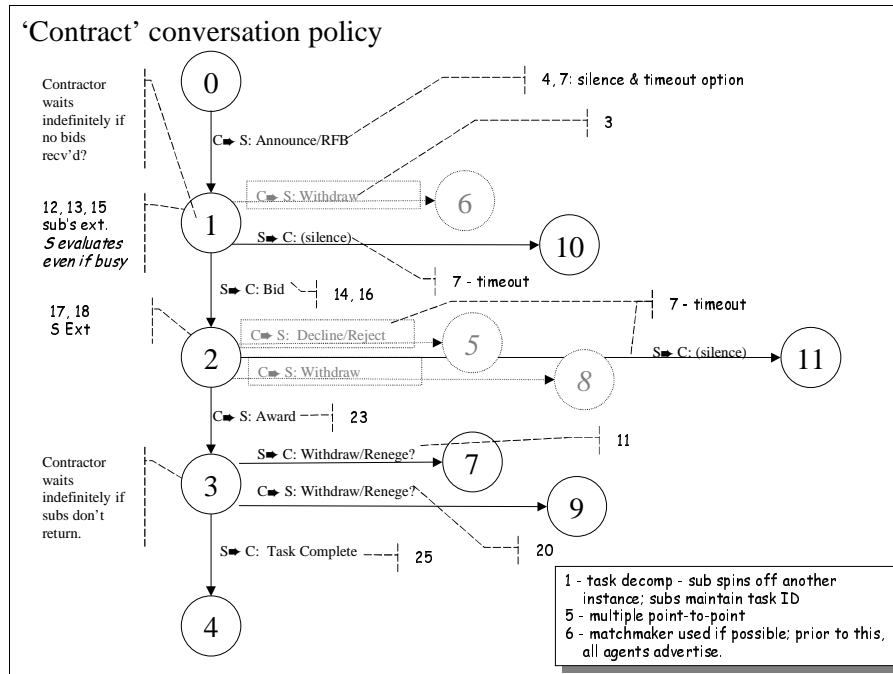


Fig. 1. A sample KAoS notation for Contract Net

lack of syntactic and semantic knowledge with which to interpret the utterance itself. These breakdowns are resolved by reestablishing the shared context of facts and inference mechanisms. However, lacking both human inferential skills and the rich shared context of human interaction, agents lack the means to directly overcome the Basic Problem in the way that humans do.

In order to address this problem, designers of ACLs like FIPA [14] and KAoS [2] have included elements in the ACL specification which further specify some of the different types of stereotypical interactions into which agents using that ACL could enter. Essentially, these elements function as templates to limit the form, content, and ordering of the possible message sequences which could be used to accomplish a communicative goal. For example, the current version of the KAoS specification of conversation types is based on a form of finite state machine (see figure 1). FIPA's interaction protocols are both somewhat more ambitious (*e.g.*, including a description of several different auction types) and considerably more vague [14]. By providing these templates, ACL designers have hoped to circumvent the Basic Problem by providing additional context information beyond that supplied by the bare form of a message. This context information at minimum includes shared expectations about the possible and required conversational moves between two agents. For example, a template might specify that a **REQUEST** will be followed by an **ACCEPT** or a **RE-**

JECT.² However, ACL designers have also included many other important types of context information in these templates, such as canonical mappings between ACL message types and the goals of the agent which generated the messages. For example, if an agent **A** knows that agent **B** sent a **INFORM** message in the context of a KAoS *INFORM* conversation policy, then **A** can conclude that **B**'s use of **INFORM** was intended to get **A** to do something – to adopt a particular belief – and furthermore that **B** expects only a limited range of replies from **A**.³ Because of the explicit policies which govern this type of KAoS conversation, **A** does not have to perform a lengthy piece of reasoning evaluating all the possible communicative goals **B** might be using the **INFORM** to achieve (*e.g.*, **B** could be using this message to indirectly get **A** to adopt a particular goal), and based on this evaluation select an appropriate response from among the universe of all of **A**'s syntactically legal responses.⁴

Unfortunately, there is no agreement in the agents community on the status of these conversation templates in the specification of agent communicative behavior, or on their standing within ACL specification practice. Different agent architectures disagree on what to call these templates (Bradshaw [2;4] uses *conversation policies*, which is the term we adopt in this paper), whether they are optional or mandatory, what formalism should be used to specify them, what conversational properties they should capture, and what their exact relationship should be to the actual message sequences that agents produce. The goal of this paper is to sort out some of these questions, and locate conversation policies within a more general model of agent communicative behavior.

2. First Principles for Conversation Policies

Our basic account of conversation policies is based on an observation that is commonplace in linguistic pragmatics and somewhat less so in ACL circles: at a fundamental level, the use of language by an agent is no different from any other action that an agent might take. Like other actions, an agent's production of a message is always the result of a plan to bring about some identified state in the world. This characteristic is the reason we use the theoretical framework of speech *acts* (and their intended *perlocutionary effects*) when we think about agent communicative behavior. In this style of theoretical framework, every agent message is driven by a strategy to achieve the agent's current goals. Of course, this process of planning and goal satisfaction may be completely implicit in any particular agent: an agent may be unreflectively executing some prebuilt procedure or state machine, and not performing any explicit deliberation at all. Nevertheless, if an agent can be coherently analyzed as having

² Note that the requirement for these responses is not a consequence of the semantics of the **REQUEST** message under most ACL semantic theories.

³ Of course, much of this reasoning might be implicit in the logic of **A**'s program, but nevertheless this is an accurate model of **A**'s ideal behavior.

⁴ For example, **B** could send an **INFORM** to **A** stating that **B** requires help on some action *a*. The KAoS conversation policy for **INFORM** forbids **B**'s goal in sending this message to be to get **A** to adopt a goal to help **B** with *a*; **B**'s direct goal in sending this message can only be to get **A** to form a belief about **B**'s capabilities for *a*.

goals and acting to bring about those goals, then our account will apply. For many researchers, the ability to take such an intentional stance is one of the prime characteristics of agenthood [1]. We implicitly adopt this stance in this paper.

Another important observation is that, unlike other types of agent actions, such as turning on a piece of machinery or booking a travel itinerary, interagent messages are fundamentally limited in the type of effects they can directly bring about. The only possible direct effect of producing an interagent message is to change the internal state of the recipient agent so that it comes to have a particular belief about the sending agent.⁵ Therefore, reasoning about the proper ACL message to generate next in a conversation necessarily involves reasoning about the beliefs, goals, and abilities of other agents, both in understanding what was intended by a received message and in projecting what the perlocutionary effects of a sent message are likely to be. Indeed, the only reason we have ACLs and conversation policies in the first place is because agents do not have direct access to the beliefs and goals of other agents. And, because reliable logical reasoning about the private beliefs and goals of others is technically extremely difficult, and especially so in light of the necessity for ACLs and the Basic Problem, implemented agent systems typically must employ various *ad hoc* simplifying assumptions.⁶ Differing simplifying assumptions about the mapping between the received ACL messages and the inferred goals of the sending agent are at the root of the agent interoperability problem outlined in the first section.

Given that agents using powerful ACLs are required to (at least implicitly) perform this type of reasoning, and given that the Basic Problem applies to using these ACLs, we can make the following broad claim:

Conversation policies are necessary in powerful ACLs in order to simplify the process of inferring another agent's relevant beliefs and goals from that agent's public messaging behavior.

After all, if it were straightforward to figure out what a sending agent intends with a message and the range of appropriate responses, then recipient agents would have no need for the additional information which conversation policies provide. This is the case in very simple and inexpressive agent communication languages, where only a limited range of interaction is possible. Agents whose interaction is bounded in this way do not require conversation policies. An example of this would be a simple agent command language in which there is only one way to express any command type. On the other hand, an agent with a high-powered reasoning engine and the luxury of time might in principle be able to entirely reason its way through a complex conversation, where every message is the result of an exhaustive analysis about its possible effects (*cf.* [12]). Such an agent might not need a conversation policy to help interpret the behavior of others. However, given that time and reasoning power are typically in short supply for contemporary agents, public conversation policies serve a critical logical function in limiting the scope of what the agent must consider [2]. Conversely, when an agent reasons about its next conversational move, public conversation policies can directly constrain the set of possible responses to consider, and (by limiting the agent's range of communicative action) indirectly limit the possible

⁵ This admittedly ignores the possibility that an agent could use an ACL message to “drown out” or otherwise make unintelligible another ACL message.

⁶ These simplifying assumptions can be seen as explicitly inducing a type of context into agent communications.

goals the agent can achieve with a response. So, we can make the following claim as well:

Conversation policies are necessary in powerful ACLs in order to simplify an agent's process of generating interagent messages that support achieving a particular set of goals.

If the two claims above are correct, we can conclude that the central role of conversation policies is to attack the Basic Problem by constraining the possible interagent messages that can appear on the wire, and in this way artificially restrict the expressive power of the ACL in use. By agreeing to use a conversation policy, an agent effectively makes public important information about how it is binding itself, and thereby makes both the reasoning and modeling task easier for all agents with which it is communicating. Specifically, conversation policies limit the possible ACL productions that an agent can employ in response to another agent, and they limit the possible goals that an agent might have when using a particular ACL expression. In the extreme case, the constraints provided by a conversation policy are so strong that the many-to-many mapping between agent states and possible ACL productions is driven down to a one-to-one mapping. That is, for a given agent goal or goal type, the conversation policies which govern an interaction will in the limiting case define a unique sequence of messages for the agents participating in the interaction to follow. In this way, the use of public conversation policies can be seen to provide a less *ad hoc* set of shared simplifying assumptions of the sort mentioned above.

We therefore view the functional role of conversation policies as publicly shared *constraints* on the potentially unbounded universe of possible semantically coherent ACL message sequences which could be used to achieve a goal. By constraining ACL usage so that the necessary communicative reasoning about beliefs and goals becomes tractable or even trivial, shared conversation policies allow agent designers to concentrate their resources on optimizing the agent's non-communicative actions – *i.e.*, the sort of actions for which we write most agents in the first place. We can state this more forcefully:

Any public declaratively-specified principle that constrains the nature and exchange of semantically coherent ACL messages between agents can be considered a conversation policy. A given agent conversation will typically be governed by several such policies simultaneously. Each policy constrains the conversation in different ways, but there is no requirement that every policy be relevant or active in every interaction.

Our account of conversation policies presents them essentially *fine-grained* – the individual constraints are presumed to only address a single feature of a particular conversation. This stands in contrast with current conversation policy description mechanisms, which attempt to regulate every relevant property of a conversation within a single policy (*e.g.*, the conversation policy shown in figure 1). We have come to believe that different conversation types should be governed by different *clusters* of policies, with some policies shared by virtually all agent conversations, such as those regulating question/answer or high level timing, and others which are applicable only to certain specific conversation types like contracting or brokering. Moreover, individual conversation policies still must take the logical power of the underlying ACL into account, because they will be closely linked to the expressive

power of that ACL. Finally, this approach to conversation policies still agents to address the Basic Problem. By agreeing in advance that a particular interaction will be bound by a given set of public constraints on ACL use, each interacting agent is able to simplify their conversational modeling task, and more easily determine the appropriate next conversational production.

Fine-grained accounts of conversation policies have an important advantage over traditional accounts. Traditional models of conversation policies (*i.e.*, transition nets) encode large numbers of individual policy decisions in a single formalism, making it difficult for both agents and agent designers to be precise about exactly which assumptions about agent communication a given model incorporates. For example, figure 1 combines policies about timing, withdrawing, sequencing, and other properties into a single complex diagram. In contrast, by conceptually separating the individual policies from the transition net computational models in which they are typically embedded, fine-grained accounts gain great flexibility about the types of policy which can be specified and the range of conversation types over which they will be valid. For example, we are now able to consider several important assumptions about the character of agent interactions that have typically been left implicit in the various extant ACL conversation policy descriptions, and separate them from the sequencing constraints which make up the traditional subject matter of a conversation policy:

1. *Termination.* It is a typical, if unstated, assumption of most agent conversation design work that agent conversations can be depended on to eventually terminate. Certain applications have stricter termination conditions, such as limits on the number of total messages that can be sent or a strict time limit on the overall length of the conversation. Other assumptions will govern the nature of acceptable termination for specific conversation types – *e.g.*, that an offer-type conversation will terminate when the offer is accepted, rejected, or withdrawn. In any case, though, there are several principles which address why individual agent conversations can come to an end, and these principles should be encoded explicitly as policies.
2. *Synchrony.* Different basic assumptions about the nature of agent conversational turn-taking, possible concurrency of messaging, interruption possibilities, and the like are often implicit in the formalisms with which current conversation policies are specified. Yet, precise knowledge of these is critical to an agent or agent designer's reasoning about how to produce messages with proper sequencing and timing.
3. *Uptake acknowledgment.* Whether or not an agent must send a message indicating successful receipt of another message, and the possible range of contents of this uptake message (*i.e.*, simple **ACK**, or some more complex message from which uptake can be inferred) is a common property of many different kinds of agent conversations. It is also a conversation policy decision, as the requirement to send an uptake messages is not typically a logical consequence of ACL message semantics. Fine-grained conversation policies can make this kind of policy explicit.
4. *Exception handling.* Exception or error conditions are always possible in real agent conversations. This means that the transition nets that have traditionally been used to specify agent conversations have had to be decorated with enormous numbers of little-used error handling transitions, often to the extent that the preferred conversational flow in the net is completely obscure. Further, many of these basic exception handling strategies are common across different types of agent

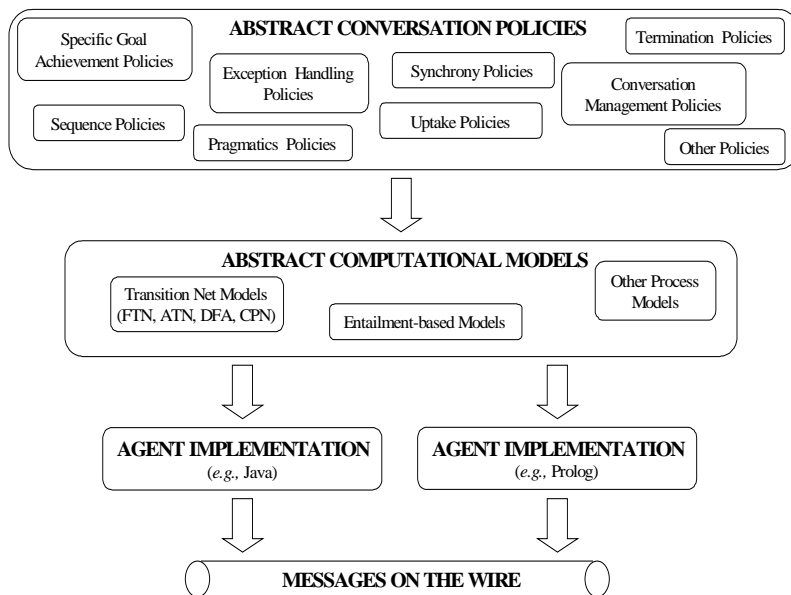


Fig. 2. A Model of Conversation Policies

- conversations. A fine-grained account of conversation policies promises to be flexible enough to account for conversational exceptions in a satisfactory way.
5. *Pragmatics.* There may be several semantically equivalent messages which can be used to achieve the same communicative goal [9;13]. Agreement between agents on the preferred way to express a given communicative act in an ACL is an important way in which conversation policies can address the Basic Problem.

This is obviously only a partial list of the kinds of implicit assumptions which regulate agent inter-action, but it illustrates our motivation for proposing an account of conversation policies focused around mixing and matching fine-grained policies in order to regulate an agent's usage of its ACL.

3. Agent Conversation Redux

Thinking about conversation policies as these sorts of fine-grained constraints on semantically coherent ACL usage is a powerful idea, and it fits well with the use of deterministic finite automata (DFA) and other formalisms which have traditionally specified conversation policies. We model agent communication with four layers, each providing a different level of abstraction (see figure 2). The direct inspiration for this approach is from Manna and Pnueli's work on reactive systems verification [ref], but this type of four-layered model of analysis is common in other parts of computer science. For example, in order to analyze how a computer performs a piece of

mathematics, we need to explain how an abstract mathematical theory (the constraints) can be reflected in one or more abstract algorithms (the computational models) and implemented in different actual programming languages and execution models (the agents).

At the top of the hierarchy is the set of conversation policies for an ACL, which are the different fine-grained constraints which agent message production in an interaction must not violate. These policies are public, shared, abstract, and normative, and can combine in different ways. When agents agree to abide by specific conversation policies for an interaction, they are agreeing to let their ACL usage be governed by a specified subset of these constraints. Further, we believe that these constraints will fall naturally into different packages: *e.g.*, general conversation management constraints, clusters of constraints relative to the use of specific communicative acts, constraints that govern message sequencing for particular kinds of goal-directed interactions, and so on. All of these constraints are sensitive to the semantic model and expressive power of the target ACL. Agent designers (or sophisticated agents), when presented with the need to interact with other agents in the service of a particular goal, will select the governing conversation policies for that interaction from the larger set of possible conversation policies for the ACL.

The set of policies that are in force for a particular agent interaction will together define a set of abstract *computational models* for that interaction. The computational models induced by a set of policies might include, for example, several different types of state transition network, but it might also range over more logically-based computational models, such as those based on entailment relations in a situation calculus or dynamic logic. The computational models are extensionally defined by the requirement that their semantics must satisfy the composition of the policy constraints that are in force; hence, these models are only normative to the extent that they accurately reflect the constraints that define them.⁷ Thus, these computational models have a type-token relationship to the actual agent conversations. Also, while policies must be shared between agents, the specific computational models induced by these policies need not be – one agent might include code implementing a DFA, while another might be running a Petri net. Nevertheless, if the agents agree on conversation policies, then the computational models used by the agents should allow them to communicate within the limits of the constraints that they agree upon.⁸ This approach allows for a wide latitude of agent implementations, and guarantees that certain properties of the conversation will hold even when sophisticated agents are communicating with simpler ones.

It is important to note that there is no requirement that the computational model chosen by an agent or designer to implement a particular set of policies regulate every aspect of an agent's messaging behavior. Models which are induced by a very tight set of policy constraints might indeed do so; but it is also possible (and indeed likely) that many aspects of messaging behavior might be unspecified by the computational model. Most commonly, computational models will provide on very general con-

⁷ In general, the choice of a computational model will fix many more properties of an agent's interaction behavior than are accounted for by the chosen set of policies. This is not a problem; presumably, agents who converse under this set of conversation policies will be able to deal with the variations.

⁸ We note that it is logically possible that agents could agree on a set of constraints that are incompatible: *i.e.*, that there is no computational model that could satisfy all the constraints.

straints on message content, and while they might include some global timing parameters, they will typically leave open the timing of individual message exchanges. More interestingly, though, a model might not specify the exact sequencing of messages required to achieve a particular state – it might specify only that a set of communicative states must be achieved in a particular sequence. This type of higher-level computational model would correspond to the agents agreeing on only a very weak set of policies to govern their interaction.

For example, a model such as the one described above might only describe a relative sequence or partial order of conversational *landmarks* (e.g., that an offer has been made; that an offer has been accepted) in a conversation of a given type. Each landmark would be characterized by a set of properties that must be true of the agents involved at that point in the conversation. Consider the *CONTRACT* conversation policy shown in figure 1. The initial segment involves an announcement made by some agent *C* immediately followed by an bid (or decline) by some other agent *S*. While it is reasonable to think of announcing and bidding as typically being a two-step process, this might not always be the case: between *C*'s announcement and *S*'s bid, *S* might ask *C* for a clarification about payment, or if a partial bid is acceptable, or whether multiple bids will be accepted. There might be any number of contract-related exchanges between *C* and *S* until the acceptance (or nonacceptance) of the original announcement. A loose set of conversation policies might not specify the number and exact sequence of messages in advance, but rather would constrain the ACL usage to attempting to achieve certain high-level states, given certain other restrictions (e.g., turn-taking).

In our view, the traditional methods for specifying conversation policies [2; 6] have *not* specified policies *per se*, but rather have attempted to specify these computational models directly, typically by using automata or transition net formalisms. The individual policy decisions that traditional conversation policies are designed to capture are only implicit in these formalisms. This has led to a significant problem in existing conversation policy design practice – how to guarantee that existing policies are interoperable or composable, how to modify them, and how to model and verify the expected effects of following these policies.

Once a set of computational models is defined, an agent designer will implement a specific choice of computational model in a particular agent for conversation of a particular type using a particular ACL. One agent might include a set of rules that simulate a DFA, while another agent whose computational model involves entailment over a particular axiom set might contain a theorem prover. A third agent might implement an interpreter for a DFA language, and download specific DFAs for a conversation from a server. The designer's choice of implementation model for an agent's communication behavior will be governed by the requirements on the sophistication of an agent's conversation, performance requirements on the agent, interaction with other parts of the agent's architecture, and so on. However, the fact that all the computational models for the conversation type will encode the same chosen constraints on ACL usage will guarantee that each agent's production of ACL messages will be limited in a consistent way. Hence, agents implementing these models will be able to minimize the Basic Problem to the extent to which their conversation policies allow.

4. Objectives for Conversation Policies

Viewing conversation policies as general constraints on semantically coherent ACL messaging behavior is a powerful idea. Besides being consistent with the general goal of addressing the Basic Problem by limiting the possible usage of the ACL, it allows us to bind together many of the classic arguments for using conversation policies to govern agent communication:

1. *Conversation policies should allow us to cleanly separate policy from mechanism in our theories of agent behavior, thus dividing public policies from private implementations.* Many authors have pointed out that the use of explicit conversation policies in agent interaction facilitates the separation of policy and mechanism – i.e., the policy must be independent of the program used to implement it. This is desirable because it enables conversation policy reuse, metareasoning over conversation policies, and off-line analysis of the implications of various conversation policy choices [8]. Viewing conversation policies as sets of interacting constraints which can induce a variety of computational models is entirely consistent with this idea, as it provides a level of analysis that is independent of the model with which the conversation policies are implemented in an agent. Explicit implementation-independent conversation policy representation also makes practical the development of consistent agent-system-wide exception handling mechanisms: unless we can unambiguously and declaratively describe expected agent behavior and the constraints by which it is bounded, we can neither detect, facilitate, nor repair problems when they arise [4; 10].
2. *Conversation policies should provide a level of abstraction that allows us to identify equivalence classes of conversations across different specification formalisms.* An important objective for any theory of conversation policies is that it provide criteria for identifying identical conversation policies across different agent implementations. If one agent is executing a theorem prover and another is running a Petri net, it should nevertheless be possible for them to have common policies governing their conversation. This means that the conversation policies themselves need to apply at a level beyond the computational models that the agents implement. Our account provides a natural way to express this.
3. *Conversation policies should be compositional, so that different pieces can be mixed and matched to govern different domain-specific conversations.* We found in our KAOs work that we were constantly tinkering with the DFAs that expressed KAOs's conversation policies. KAOs conversations have certain common features, such as the way that question/answer loops and exceptions are handed, and we wanted to be able to flexibly integrate these with our existing DFAs. However, without a precise way to describe the critical properties of the original DFAs that we needed to preserve, we could not guarantee that the modified DFAs expressed the same ideas about conversation design as the original ones. Policies expressing certain kinds of pragmatic constraints were difficult to express as pure DFAs (e.g., timing, abnormal termination, and general turn-taking constraints) and we found ourselves extending the representation in ad hoc ways to reflect these concerns

rather than applying a more systematic approach. Essentially, we found that the nature of the conversations we were designing required fine-grained policies that could be combined to yield larger composite policy entities. The present framework allows us to do that.

4. *Conversation policies should be flexible enough to allow agents of different levels of sophistication to interoperate, and should therefore allow for conversational control at different levels.* The successful agent-based ensembles of the future will include agents created by different vendors with widely varying degrees of sophistication and reasoning ability, operating under many different kinds of resource constraints, and interacting with each other at many different levels [3;4]. Many agents will be small and simple, some will have medium-scale reasoning abilities, and relatively few will exhibit complex and explicit reasoning on beliefs and goals. All will have to communicate with each other, at least to advertise their services and autonomously negotiate for resources they need. This kind of extreme heterogeneity entails that agents will need to tailor their use of the ACL to match the capabilities of the interacting agents – the highly restricted language necessary to interact with a simple database agent wrapper is too inexpressive to handle, e.g. complex distributed planning problems. By expressing individual ACL constraints at a level above the composite computational models, agents and agent designers can reason about how precisely to manage the ACL tradeoffs that the Basic Problem requires, and match these tradeoffs to the capabilities of the agents that are conversing.

5. Conclusion

We realize that the approach to conversation policies sketched out in the above text will require a great deal of formal work to make it precise and convincing. We are currently engaged in research which we believe will do just that. Specifically, we are looking at:

1. Formal languages for representing conversational constraints and computational models
2. Deductive techniques which will allow us to verify that a computational model meets a specification.
3. The relationship and boundaries between ACL semantic theories and the account of conversation policies given here.

Though agent designers who have not been dealing with conversations in any formal way may worry that the kinds of approaches discussed in this paper will just add unnecessary complexity to their implementations, we assert precisely the contrary. We believe that the combination of powerful offline conversation analysis tools, with the resultant systematically constructed conversation policies, policy facilitators, enforcement mechanisms, and exception handling facilities will actually simplify the online reasoning task for agents and help guarantee a level of robustness and responsiveness to pragmatic considerations that has been lacking in deployed agent systems

[4].

Our hope is to move the analysis of agent conversations up a level, and make explicit all of the fine-grained policies which have hitherto been implicit in the conversations we have designed. In this way, the view that we are advocating is not substantially different from many other types of formally-based scientific work. It is always important to bring out the relations between the abstract theories which provide our intellectual touchstones, the process models which allow us to combine our theories into constructive procedures, the artifacts in which our theories are implemented, and the results of allowing these artifacts to interact in the world. Our account of conversation policies is a step in this direction.⁹

6. Acknowledgments

The authors thankfully acknowledge support from the DARPA CoABS program (Contract F30602-98-C-0170); the Aviation Extranet joint-sponsored research agreement between NASA Ames, The Boeing Company, and the University of West Florida (Contract NCA2-2005); and the Agency for Health Care Policy Research (Grant R01HS09407).

References

1. Bradshaw, J. M. (1997). An introduction to software agents. In J. M. Bradshaw (Ed.), *Software Agents*. pp. 3-46. Cambridge, MA: AAAI Press/The MIT Press.
2. Bradshaw, J. M., Dutfield, S., Benoit, P., and Woolley, J. D. (1997). KAoS: Toward an industrial-strength generic agent architecture. In J. M. Bradshaw (Ed.), *Software Agents*. pp. 375-418. Cambridge, MA: AAAI Press/The MIT Press.
3. Bradshaw, J. M., Gawdiak, Y., Cañas, A., Carpenter, R., Chen, J., Cranfill, R., Gibson, J., Hubbard, K., Jeffers, R., Kerstetter, M., Mathé, N., Poblete, L., Robinson, T., Sun, A., Suri, N., Wolfe, S., and Bichindaritz, I. (1999). Extranet applications of software agents. *ACM Interactions*. Forthcoming.
4. Bradshaw, J. M., Greaves, M., Holmback, H., Jansen, W., Karygiannis, T., Silverman, B., Suri, N., and Wong, A. (1999). Agents for the masses: Is it possible to make development of sophisticated agents simple enough to be practical? *IEEE Intelligent Systems* (v. 14:2, March 1999), pp. 53-63.
5. Cohen, P. R., and Levesque, H. (1997). Communicative actions for artificial agents. In J. M. Bradshaw (ed.), *Software Agents*. pp. 419-436. Cambridge, MA: The AAAI Press/The MIT Press.
6. Finin, T., Labrou, Y., and Mayfield, J. (1997). KQML as an agent communication language. In J. M. Bradshaw (ed.), *Software Agents*. pp. 291-316. Cambridge, MA: The AAAI Press/The MIT Press.
7. Greaves, M., Holmback, H., and Bradshaw, J. M. (1999). Agent conversation policies. In J. M. Bradshaw (ed.), *Handbook of Agent Technology*. Cambridge, MA: AAAI Press/The MIT Press. Forthcoming.

⁹ We are currently working on a more complete account of conversation policies to appear in [7].

8. Greaves, M. T., Holmback, H. K., and Bradshaw, J. M. (1998). CDT: A tool for agent conversation design. *Proceedings of 1998 National Conference on Artificial Intelligence (AAAI-98) Workshop on Software Tools for Developing Agents*. pp. 83-88. Madison, WI, Menlo Park, CA: AAAI Press.
9. Holmback, H., Greaves, M., and Bradshaw, J. M. (1999). A pragmatic principle for agent communication. J. M. Bradshaw, O. Etzioni, and J. Mueller (ed.), *Proceedings of Autonomous Agents '99* Seattle, WA. New York: ACM Press. Forthcoming.
10. Klein, M., and Dellarocas, C. (1999). Exception handling in agent systems. J. M. Bradshaw, O. Etzioni, and J. Mueller (ed.), *Proceedings of Autonomous Agents '99*, Seattle, WA. New York: ACM Press. Forthcoming.
11. Manna, Z. and Pnueli, A. (1995). *Temporal Verification of Reactive Systems: Safety*. New York: Springer-Verlag.
12. Sadek, M. D., Bretier, P., and Panaget, F. (1997). Artimis: Natural Dialogue Meets Rational Agency. *Proceedings of the 1997 International Joint Conference on Artificial Intelligence (IJCAI-97)*, Palo Alto: Morgan Kaufmann.
13. Smith, I. A., Cohen, P. R., Bradshaw, J. M., Greaves, M., and Holmback, H. (1998). Designing conversation policies using joint intention theory. *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS-98)*. pp. 269-276. Paris, France. Los Alamitos, CA: IEEE Computer Society.
14. Steiner, D., (ed.) *FIPA 97 Specification Version 2, Part 2: Agent Communication*. <http://www.fipa.org/spec/FIPA97.html>.