

A foundation for flexible automated electronic communication

Scott A. Moore
University of Michigan Business School
samoore@umich.edu

January 29, 2000

Abstract

In this paper the author describes a formal language for communication based on linguistics—more specifically, a theory of natural language communication and models of natural language conversations. The language has a small number of general message types which are formally defined by their intended effects on the recipient. For each message type he defines a standard, automated method of responding that depends only on the message type and is independent of the message's content. For more complex conversations he provides methods for responding that do depend on the content. In this system, a message's sender—automated or human—constructs and sends a message knowing that he cannot know, but can only predict, how it will be interpreted. The agent receiving the message interprets it and then uses it as a basis for inferring how it should respond. The message interpretation mechanism for this language is reusable, modular and shared by all applications. The benefit of this communication system is that it makes the communication infrastructure more flexible, easier to modify, easier to expand, and more capable.

1 Introduction

With the current stampede toward electronic commerce, businesses no longer have the luxury of setting up trading-partner-specific communication systems. Businesses want to communicate with and do business with people and companies with whom they have not had any prior contact. They also want

to build more extensive linkages with their best trading partners and in their outsourcing relationships. Companies have built these links using electronic data interchange (EDI). Unfortunately, EDI cannot be implemented easily, quickly, or inexpensively and is therefore inappropriate for many companies and people to adopt.

The technologies proposed in this paper are vital for moving into the new world of Web-based electronic commerce. Consider a situation in which *Company A* is a supplier for *Company B*. *Company B* routinely asks *Company A* about its production schedule while *Company A* asks *Company B* about its inventory levels. They decide to set up their computer systems so that they can exchange EDI messages. Even if these companies use standards-based EDI, the latitude allowed by the standards requires that the companies agree on how the EDI messages will be structured and interpreted. They also sign a business contract covering these exchanges. *Company B* later decides to exchange EDI messages with another company. It will have to go through another round of signing an agreement, configuring its system, and agreeing on interpretations. It would attempt to push its structures and interpretations onto this new company. If it is unsuccessful, then it will have to incorporate these alternatives into its systems—hoping that they do not affect *Company A*'s messages. Overall, this process is inflexible and difficult and expensive to maintain.

Web-based commerce does not allow for such a long wind-up period before the delivery of the benefits. Companies want to simply press a button and send a message. They do not want their topic of conversation to be limited by the EDI standard's current status, they do not want their conversational partners to be limited by legal contracts, and they do not want the speed at which changes can be made to be limited by the speed at which large changes can be made to their communication system. Certainly e-mail and telephones and face-to-face meetings—and, possibly, in the future, chat rooms, MOOs and MUDs—provide flexible communication avenues for meeting and discussing business. However, the cost of this flexibility is that people have to be intimately involved in the communication process—slowing the whole process down and introducing inconsistent message handling. As Web-based forms and automated mailing list management have demonstrated, sometimes a person is not needed on the other end. And, as EDI has demonstrated, sometimes people are not needed on *either* end. However, what EDI does not provide is the ability to add new messages and conversation partners easily.

An application based on the technologies proposed in this paper would have, already built in, a certain capability for understanding messages plus a

flexibility that would allow the system to grow with few restrictions provided by the underlying communication system. The benefit is that almost any message that is structured correctly can be handled to some extent; further, more complex message handling for new messages can be defined incrementally. Plus, by separating the message from the production of effects on the speaker (to be discussed later), electronic communication with unknown parties—e.g., Web-based e-commerce—becomes much safer. The cumulative effect of these benefits is that such a communication system would be both easier to implement and more flexible.

The same attributes that make these technologies attractive as a replacement for or an adjunct to EDI make it equally attractive as an agent communication language (ACL). In both EDI and ACL a company or agent wants to exchange messages with as many other partners as possible about as many topics as possible with as little preparation as possible. The set of topics and partners should be able to change and expand quickly. The communication system discussed in this paper decreases the amount of programming effort that has to be expended (compared to current agent communication languages) both for sending a new message between partners and for exchanging messages with a new partner (see Moore [44]).

This paper describes in detail an approach to computer-processable messaging. Kimbrough & Moore have proposed [25, 27, 29, 43] that a more expressive language and a more complex model of communication are needed in order to make EDI systems easier both to develop and to extend. In this paper I describe a language (the FLBC, discussed in §4.1) and a new system for interpreting messages (the MMS, §4.2) which are both based on a linguistic theory called speech act theory (SAT, §3.1). The purpose of basing an automated communication system on a linguistic theory is so the system can gain the benefits of natural language communication, namely, range of expression (i.e., exchange messages... about as many topics as possible) and ease of adaptability (minimal preparations). In this way it is similar to McCarthy's work relating SAT to programming languages (Elephant 2000 [41]) though it differs in its emphasis on communication instead of on program specification. Prototypes based on a less-developed version of this theory have been quite encouraging. For example, both an office communications system and a system that provided inter-organization communication for a supply chain successfully demonstrated the flexibility of applications employing this theory [46]. The language also proved flexible enough to encode messages from the EDIFACT EDI standard, a proprietary financial services EDI standard, and the inter-application communication standard on the Apple Macintosh [43].

The language described in this paper has message types that are formally defined by their *intended* effects on the recipient. In this system, a message's sender—automated or human—constructs and sends a message knowing that he¹ cannot know, but can only predict, how it will be interpreted. The agent receiving the message interprets it and then uses it as a basis for inferring how he should respond. Systems that use this language have a modular message interpretation mechanism called the MMS (§4.2). Appropriate applications of this technology include any system that needs to send computer-processable messages to another system. It is especially applicable for those in which development is distributed among many organizations because the message interpretation process explicitly recognizes uncertainty and issues of trust that surround communication. Thus, this system could be used to support agents representing buyers and sellers in a marketplace, opposing sides in a negotiation, vendors and customers in a supply chain managing the delivery of goods, a car notifying its owner's digital assistant that repairs are needed, or a person creating a message with a form on a Web page as he reports trouble at a customer service site.

2 Motivation

The problem addressed by this paper may not seem to be much of a problem anymore. After all, many solutions exist for enabling applications on different computers to talk with each other. TCP/IP and the Internet each make it easier (at different levels) for applications on different computers to send bits back and forth. Other, higher-level, specifications have also addressed this problem. CORBA [48] allows programs to invoke methods on a server object, independent of the object's location, programming language, or operating system. This is convenient, but does not address the problem at hand which concerns the most appropriate structure for communications. Certainly, a message might be deposited on another computer by invoking a CORBA object, but the question remains: what should the structure of that message be? Neither is remote method invocation, of which CORBA is an instance, a perfect substitute for inter-application messaging. It is not always desirable to allow your application to be called by programs deployed and controlled by developers from other companies, either for security or computer-load management reasons. Another related technology is XML (see discussion in Appendix A). It provides an easily parsed syntax—it

¹I'll use "he" instead of "it or he or she" for conciseness though the latter is more accurate.

System	Refers to SAT	Tagged	Fully formal	Formal definition	General message classification	Recursive content	Formal message interpretation
Coordinator [18, 55]	■	■			■		
Information Lens [40]		■					
KQML [31, 33]	■		■	■			
COOL [7, 8, 9]	■		■				
Agent-0 [53]			■				
FIPA ACL [19]	■		■	■	■	■	
FLBC	■		■	■	■	■	■

Table 1: Communication systems

looks like HTML—which can be used to define a fully computer processable and interpretable message. Again, this technology does not tell us what the structure of a message should be; it merely allows language designers to structure messages in any way they see fit.

Neither CORBA nor XML obviates the need for a computer-processable language, many of which have recently been defined as agent communication languages, the most active area of the study of these languages. However, many already exist so it is probably unclear as to why another one might be of interest and why I might be proposing one here. Consider Table 1 in which I classify some earlier, prominent systems that sent at least partially formal messages by whether or not the system’s underlying theory refers to speech act theory (SAT), whether the messages are tagged or fully formal, whether or not the message definitions are formal, whether or not the message classification is meant to be general, and whether or not the message can be recursively formed. The meaning of the first column is explained later in this section. A tagged message is one in which the message is classified according to some scheme but the content is expressed separately from the classification as text that cannot be automatically processed. A fully formal message can be accurately and easily interpreted by a computer. Email is an example of informal messaging in which the computer has the contents of the message available to it but is not able to interpret the contents in this manner. A language which formally defines its messages contrasts with those whose definitions are simply provided in English text with all its subtleties

and nuances. The formality helps ensure that the definitions are interpreted consistently across organizations. A language whose message classification is claimed to be general should need only infrequent changes to its set of message types. Finally, a message that can have recursive (or iterated or composed) content is one in which a message can have content composed of another message. This freedom allows the creation of messages that are, for example, a “request to predict” without defining a new message type. Moore has shown that EDI messages [43] and ACL messages [44] frequently contain recursively defined messages. The ability to express this type of message makes it much easier to express a huge range of new messages. Finally, a formal model of the message interpretation process defines a standard way for systems to process messages while also defining a standard set of errors that will be caught by the system. This also helps ensure that messages are interpreted consistently from one organization to another.

Thus, what is *not* available is a language whose messages are fully formal, whose meaning is formally defined and implementable, that has a message classification whose set of types does not have to be frequently extended, that allows recursively-defined content, and that has a formal model of the message interpretation process. This type of language would meet the needs of organizations that want an EDI application or an agent-based system that can communicate about new topics with new conversation partners with a minimal amount of effort. Consider the systems in Table 1. Since both the Coordinator and Information Lens are systems for sending tagged messages, neither is appropriate for current purposes. KQML is the *de facto* standard ACL. Several authors, notably Covington [15], have addressed KQML’s shortcomings in detail. Here I will merely focus on one difference between the languages. While FLBC’s set of message types has been stable, KQML is rather open-minded in its attitude toward these performatives (its name for message types): “Though there is a predefined set of reserved performatives, it is neither a minimal required set nor a closed one. A KQML agent may choose to handle only a few (perhaps one or two) performatives. The set is extensible; a community of agents may choose to use additional performatives if they agree on their interpretation and the protocol associated with each.” [17, p. 460] This leads to the not unexpected result that agents developed by different communities of developers generally cannot communicate. This is clearly not attractive given the necessity of distributed development in large e-commerce related applications. COOL is an extension of KQML. Though it has many useful features especially relating to conversation management, it suffers from the same shortcomings as KQML. **Agent-0** is a relatively minimalist language (only `requests`, `informs`, and `unrequests`

are allowed), but its modeling of an agent's beliefs and commitments are complex. This is a fully-formal language but provides neither for separation of type-specific effects from content-specific effects nor for a formal definition of the meaning of its message types. ACL, the Agent Communication Language proposed by FIPA, is the newest, and perhaps most complete and complex entrant into the field. It differs from FLBC in two ways. First, its semantics are so complex that it is doubtful that a system could actually implement them accurately anytime soon. Second, it does not have anything that corresponds to the MMS's formal model of message interpretation (i.e., the FL-SAS, discussed in §4.2).

It is not at all clear what a researcher must do if he wants to define a language. First, he must define a set of terms that can be used in a message (the vocabulary) and a means of putting those terms together to form a valid message (the grammar). The size of the vocabulary partially reflects the number of objects and concepts in the application area; the language developer cannot exert much control over this. The complexity of the grammar increases relative to the flexibility of the ordering of the terms, the number of terms whose inclusion in the message is optional, and the number of parts the message might have. The complexity of the grammar has a lot to do with how difficult it is to create messages that say what is needed and then to build applications to reliably interpret them.

The complexity of the world that these messages are meant to function in will be reflected in the complexity of the grammar, the number of message types, and vocabulary. To make the point clearly, think of two different languages: the first with two message types, assert and request, and the second with thousands of message types. The complexity of the first language clearly is not in the message types, so if the messages are going to discuss a complex world, then the language will have to have a complex vocabulary. The second language clearly makes the opposite choice: create a separate message type frequently and attempt to minimize the changes made to the vocabulary. These languages would be appropriate for different applications. Below I discuss several systems and the choices that were made in implementing them.

One choice for distributing complexity is to define a moderately complex grammar with a huge and growing vocabulary—e.g., natural language. Though this allows the speaker to express just about anything that he needs to say, it is difficult to interpret. Another choice is define a separate grammar for each different type of message. The idea usually is to specify as clearly and unambiguously as possible the terms that can be used in that message. An example is EDI. Each type of message defines its own struc-

ture (i.e., grammar). Though the components of the message are basically re-used among messages, the outer structure and how each component is interpreted are unique to each message. This choice can simplify the interpretation process since ordering is well specified. Even though EDI uses this approach, optional terms are prevalent and the number of parts is high, both of which increase the complexity of the interpretation process.

The approach typified by EDI finds favor with many people but it creates much difficulty in practice. Kimbrough & Moore [30] examined EDIFACT messages Compaq uses with their business partners [14] in an attempt to determine the semantic content of EDI messages used by industry. The documentation provided by Compaq is meant to be used by trading partners who are implementing their own EDI system. Their analysis found many instances in the guidelines in which Compaq's guidelines differed from the standard or the standard was so unclear that the meaning of fields or even whole messages remained ambiguous even with Compaq's guidelines. For example, they point out that it is not clear from the guidelines or from the EDIFACT standards if the `ORDRSP` document is only able to confirm the order to which it is responding or if it is able to modify the order. To clarify these misunderstandings, Compaq and each trading partner must come to agreements on them either in a formal interchange agreement or informally by practice. It is this clarification process and the level of minute detail at which it must be performed that add many months to the process of adding an EDI partner and that makes it unsuitable for current purposes.

Another choice is to have a fixed message structure and a small set of message types. The fixed message structure means that a single message interpretation procedure can be applied to messages of all types. This contrasts strongly with EDI. To handle complexity, the vocabulary must be rich and the rules for combining the terms must be flexible. The difficulty with this approach is figuring out what that universal message structure might be. In some branches of linguistics as well as nearly universally in computer science it is common to use a functional form (e.g., $f(x)$). The f can be thought of as defining the message's type. Applying this to messages the researcher would have to come up with a classification system for different types of statements—that is, come up with the set of functions. Difficult questions remain: What types should be used? How many should there be? How complete should the list be? Should it be expandable?

One way to approach these questions is to look at the messages the system sends or might send. Though this is a reasonable place to start, there is a fairly good chance that a message considered after the construction of the original set might require a new type of message. If new messages are

commonly needed, then developers would have to make frequent changes to the basic message interpretation mechanism. This would not be good. An alternative is to look at theories that might have an ontology relevant to the application area in which the system will be used. Since the application area for an agent communication language is essentially anything that might be communicated, a general theory of communication would be appropriate here. This is, in fact, the approach some researchers have chosen, including me with my reference to speech act theory.

Now, suppose that the researcher has defined the vocabulary, the grammar, and the message types. The next step is to ensure that messages are interpreted consistently by agents that use this language. This is *the core problem* of defining a language used for automated communication. Though it is somewhat difficult to define a language, a researcher can arrive at a reasonable definition. What really distinguishes one language from another is the approach the researcher takes to the problem of consistent interpretation. This has not always been so—the Internet has increased its importance. Formerly researchers did not necessarily address it explicitly because a small team of researchers worked closely together to implement all the senders and receivers of messages in a particular language. The implementor of one agent could simply communicate face-to-face with that of another and resolve any differences. With the spread of Internet based electronic commerce this simply is no longer possible. So, now, other approaches must be used to address this problem.

One approach is to define a formal specification of the automated conversants who use the language. This specification would define in terms of the receiving agent’s capabilities the effects that the message would have. This is a common and useful approach when the researcher can assume that all communicating agents share a set of capabilities. When the domain is Internet based electronic commerce and agents are developed by researchers world-wide, it is hard to make this assumption.

Another approach is to define an informal model addressed to the developers themselves. This is essentially what the developers of KQML did with their first specification [16]. They wrote English language descriptions of how they intended the message to be interpreted. This worked as long as the community of KQML developers was small but proved unworkable as it grew. KQML *is* widely implemented; however, agents developed for a particular research project generally do not interoperate with those from another one because each team makes slight additions and changes to the interpretation of the messages. These changes do not necessarily conflict with the specification. The specification is simply open to varied interpreta-

tions that eventually leads to messages that have unforeseen and unwanted consequences.

A third approach focuses on defining mostly formal models for developers. This approach is the one used in this paper. This provides an unambiguous specification for developers, and should lead to a greater level of interoperability among agents.

What I do in this paper is provide a formal definition, aimed at developers, of a language and its associated message interpretation mechanism. These systems form a communication infrastructure (whose features I describe above) for a broad class of applications. Much of the generality of this language comes from its reliance on linguistic theories, and it is to that topic that I turn now.

3 Background

The communication system described in this paper is based on the premise that, by applying what we know about natural language communication, we can improve computer-based automated communications systems. Improvements are expected to come in the increased variety of messages that can be easily handled. Foundational theories for this system include speech act theory (§3.1) and models of natural language conversations, also known as discourse or dialogues (§3.2). The following section presents a fairly complete explanation of how communication works from a SAT perspective. The purpose of this presentation is to give the reader both an introduction to the SAT approach to communication and a means by which to evaluate the description of the communication system that follows in the next section.

A disclaimer: I am *not* saying that this is exactly how human communication works. In fact, my argument does not even rest on the accuracy of these theories. I am merely proposing that the description of communication as outlined by these theories provides a model for a flexible automated communication structure. This investigation is aimed at determining both how much can be feasibly implemented and what the benefits are.

3.1 Speech act theory

Both the formal language for communication and the theory developed to respond to messages in that language—soon to be described—are partly based on speech act theory (SAT). In the following I describe those features of SAT that are material to the purpose at hand.

Recent work in linguistics and philosophy of language—aimed at developing theories of how language understanding and communication works—has emphasized the role of inference and context (e.g. [4, 36]). In concert with this work, a theory of linguistic communication has been developed by linguists, philosophers, psychologists, and cognitive scientists generally. (Of course, there is precursor work, particularly [54].) As noted above, this theoretical approach is called speech act theory. This theory is so named because its adherents assert that to say something is to perform an act. *Act* is being used in a technical and theory-laden, if not altogether clear, way. Briefly, to act is to do something with an intention. Falling down is usually not an action, pulling a lever in a polling booth normally is. Although there is no generally accepted full description of the theory, two core ideas are accepted by speech act theorists. These two core ideas prove useful for developing a formal language for business communication.

The first core idea of SAT is every (or nearly every) speech act is the speaker’s expression of a propositional attitude toward some possibly complex proposition. For example, if the speaker says “It will rain,” then typically the speaker is predicting that it will rain. The proposition is that it will rain and the propositional attitude is that of a prediction. If the speaker says, “Will it rain?” then typically the speaker is asking whether it will rain. In this case, the proposition is the same—that it will rain—and the propositional attitude expressed is that of a question. Thus, different attitudes can be expressed toward the same proposition. Speech act theorists call these attitudes *illocutionary forces*. This first core idea is summarized by saying that every speech act has the structure $F(P)$, where F is an illocutionary force applied to a proposition, P , called the *propositional content* of the act. This is called the $F(P)$ *framework*. This strongly influences the formal language: if every utterance (or message, or *speech act*) has the $F(P)$ structure, then the formal message, no matter what it is expressing, must represent both the illocutionary force and the proposition. No other message formats are necessary.

The second core idea of SAT is the idea that every speech act consists of several distinct actions. The idea and most of the terminology come from Austin [3]. Recognizing that different authors distinguish differently among the various constituent acts and even recognize different acts, for present purposes a speech act is representable by four distinct actions. Suppose that a speaker, s , succeeds in saying something to a hearer, h , in a given context, c . For now think informally of the *context* as those elements of the message that are not directly captured in the utterance. The following acts can be distinguished:

utterance act the uttering of *u* by *s* to *h* in *c* of a particular expression from a given language.

locutionary act the actual saying of something by *s* to *h* in *c*.

illocutionary act the doing of something by *s* in *c*, in virtue of having performed the utterance act.

perlocutionary act *h*'s being affected by *s* in *c*, in virtue of *s*'s utterance act.

The general picture of communication and understanding that emerges is this. A linguistic communication—a successful *speech act*—may be viewed as a sequence of the above four acts, which (after [4]) I call the *speech act scenario* (SAS). Bach & Harnish have described an SAS as a series of inferential steps that a hearer goes through in his attempt to understand what a speaker says; see Figure 1. The following list demonstrates what their SAS says about the hearer's inferential steps in response to a specific utterance (and also explains the notation used in the figure).

1. Steve, handing a letter to Scott, emits some sounds (*e*) that can be represented by [pli:z me:l ðis letr tu: meri:]. These sounds are the *e* and this is the utterance act. Scott has assigned no meaning yet to *e*—it is merely a series of sound waves.
2. Scott reasons that Steve means to say “Please mail this letter to Mary” (this is the “...” in the SAS). He has recognized the sounds as a statement in English and recognized them as words in a sentence. The utterance act has apparently succeeded.
3. Scott reasons that Steve is saying that Steve is asking Scott to mail the letter (that he is handing to Scott) to Mary (in Atlanta—the only Mary that Scott believes that they both know). Scott has clarified Steve's external references and any ambiguities in the utterance. The **(...p...)* form “indicate[s] that what is said is a function of the intended meaning (“...”) of the expression *e*” [4, p. 9]. The locutionary act has apparently succeeded.
4. Scott reasons that Steve, if speaking literally, is requesting (*F**-ing; that is, the illocutionary force is *F**, *request* in this case) that Scott mail the letter that he is handing to Scott to Mary in Atlanta (*p*). It is the case that the *p*—the propositional content—is something Scott

<ol style="list-style-type: none"> 1. S is uttering e. 2. S means ... by e. 3. S is saying that *(... p ...). 4. S, if speaking literally, is F*-ing that p. Either (<i>direct literal</i>), 5. S could be F*-ing that p. 6. S is F*-ing that p. And possibly (<i>literally based indirect</i>) 7. S could not be merely F*-ing that p. 8. There is some F-ing that P connected in a way identifiable under the circumstances to F*-ing that p, such that in F*-ing that p, S could also be F-ing that P. 9. S is F*-ing that p and thereby F-ing that P. 	<p>Or (<i>direct nonliteral</i>),</p> <ol style="list-style-type: none"> 5'. S could not (under the circumstances) be F*-ing that p. 6'. Under the circumstances there is a certain recognizable relation R between saying that p and some F-ing that P, such that S could be F-ing that P. 7'. S is F-ing that P. And possibly (<i>nonliterally based indirect</i>) 8'. S could not merely be F-ing that P. 9'. There is some F'-ing that Q connected in a way identifiable under the circumstances to F-ing that P, such that in F-ing that P, S could also be F'-ing that Q. 10'. S is F-ing that P and thereby F'-ing that Q.
--	---

Underlying assumptions (these are paraphrased):

Linguistic presumption. The members of the linguistic community share a common language. If a speaker speaks clearly, then the hearer can know the meaning of the utterance. [4, p. 7]

Communicative presumption. When a speaker says something to a hearer, he has a recognizable intent for doing so. [4, p. 7]

Presumption of literalness. When a speaker speaks, it is mutually believed that if he could be speaking literally, then he is speaking literally. [4, p. 12]

Mutual contextual beliefs Beliefs “relevant to and activated by the context of the utterance (or by the utterance itself)” [4, p. 5] that are held by both the speaker and hearer and which both believe “that they both have them and believe the other to believe they both have them” [4, p. 5].

Figure 1: Bach & Harnish's speech act scenario [4, pp. 76–7]

can do or can be responsible for doing. This corresponds to Bach & Harnish's concept of being *locutionary-compatible*. For a person to be speaking literally and, in doing so, to be predicting something, that something has to be in the future for it to be locutionary compatible with a prediction. For a person to be speaking literally and to be speaking of something that has already occurred, then the statement is not locutionary compatible with a prediction. Scott has formed a hypothesis that Steve is speaking literally; that is, the meaning of the utterance (what was accomplished in step 2) delimits the illocutionary acts which Steve could possibly accomplish. Checking for locutionary compatibility is part of the process the hearer goes through in validating this hypothesis.

5. Scott thinks that it is reasonable, given the current circumstances, to think that Steve is requesting that Scott mail the letter to Mary.
6. Steve is requesting that Scott mail the letter to Mary. The illocutionary act has apparently succeeded.

Scott now recognizes Steve's intention and so it can be said that the speaker's illocutionary act succeeded. The success of this act does not depend on Mary receiving the letter, Scott mailing the correct letter, or even Scott mailing *any* letter to Mary. The success of the communicative act depends solely on the hearer recognizing the intention of the speaker—in this case, that a particular letter be mailed by Scott to a particular person. The SAS has nothing to say about how the hearer *responds* to an utterance, only about the hearer recognizing what the speaker meant to say by the utterance.

The rest of the SAS describes communication that is neither direct nor literal. For purposes of this article, I have assumed that computer-based formal communication must be both direct and literal. For this reason I will ignore these portions of the SAS in the rest of this paper. If need be, these could be incorporated into future research since nothing done in this research would prohibit it.

The perlocutionary act is the follow-on to the communicative act. It consists of the actions that are an intentional result of the success of the illocutionary act. In this instance, it could be that, as a result of the hearer recognizing the speaker's intention, the hearer mails the letter to Mary. This action is the perlocutionary effect of the communicative act. Further, it is the intended effect of this particular communicative act. Another effect might have been to remind Scott that he has some letters at home that he

assert In uttering e , S asserts that C if S expresses:

1. the belief that C , and
 2. the intention that H believe that C .
-

Figure 2: Bach & Harnish’s definition of *assert*

needs to mail; this would not be a perlocutionary effect because Steve did not intend for this to happen.

The explication above glossed over both the range of illocutionary forces and the meaning of each. Bach & Harnish define a set of about thirty illocutionary forces, twenty-two of which are shown in Appendix B. I list *deny* as a separate force though Bach & Harnish list it as a special type of *assert*, specifically, an assertion that *not P*. This definition contrasts with that of *dispute* which is an attempt by the speaker to get the hearer to *not believe* that P .

The *request* and *question* forces are similar enough to cause some confusion. A *request* occurs when the recipient believes the speaker wants the recipient to do something. This “something” might be an action, it might be to send a message, it might be to answer a question about a database, etc. A *question* is a specific type of request; it occurs when the recipient believes the speaker wants the recipient to do two specific things: determine the truth of a statement, and then inform the original questioner of that truth value. Other types of questions (not the force *questions* but relating to questions people ask each other) can be asked by using different content within a message with force *request*.

I do not claim that these are the only illocutionary forces, or right ones, or even the best ones—I merely claim that they provide a good place to start exploring. A bit more strongly: Moore showed [43] that UN-EDIFACT EDI transaction sets, an industry-standard set of EDI messages for the financial industry, and standard AppleEvents (IAC messages on the MacOS) can all be classified within this set of illocutionary forces. Thus, this set is a reasonable place to start.

Each illocutionary force has a detailed definition. Consider the definition for *assert* in Figure 2. If the speaker’s illocutionary act succeeds after he has asserted C —that is, if the speaker has spoken literally and directly, then the hearer has gone through steps 1–6 of the SAS—then the hearer will recognize that the speaker has, in fact, asserted C . The success of an illocutionary act rests on the hearer recognizing the act as that act which the speaker meant

for the hearer to recognize—that is, the hearer correctly recognizes the intent of the speaker. This is a “reflexive intention,” in the spirit of Grice [21]. Given that the hearer has recognized the speaker’s intent, then the hearer also recognizes what the speaker means for him to believe. Specifically, for “assert” this entails (again, see Figure 2) that the hearer recognizes that the speaker means for the hearer to believe that 1) the speaker believes that C, and 2) the speaker wants the hearer to believe that C. Neither of these determine *how* the hearer responds to the message, the ultimate beliefs of the hearer, nor the ultimate effects this utterance has on the hearer. Both indicate that the hearer has recognized what the speaker means the effects of the statement to be. Both are beliefs the hearer has about the speaker. The first is a belief about the speaker’s beliefs; the second is a belief about the speaker’s intentions. These effects change what the hearer believes about the speaker but do not (yet, or even necessarily) change what the hearer believes about himself. The other illocutionary forces are defined similarly.

I propose (and later describe how in detail) that the perlocutionary act—that is, the responses the hearer has to the speaker’s utterance—be broken into two parts. Other researchers have neither made this distinction nor seen the benefits of doing so. The “standard effects” occur as a result of the success of the illocutionary act. These are the effects described just above for *assert*. (All the forces are defined in Appendix B.) Standard effects can be defined without knowing the message’s content. The extended effects, if any, are inferred after the standard effects. These effects will be brought about because of changes in the hearer’s beliefs and goals. These effects go beyond what is predicted given the definition of the illocutionary force. For an assertion this might be that the hearer does something because of the change in beliefs resulting from the assertion. These effects depend on the meaning of the message, the recipient of the message, and the recipient’s current beliefs and goals.

3.2 Models of discourse

SAT does not address the problem of modeling discourse (i.e., conversations). Certainly, engaging in a conversation requires that utterances be made and illocutionary acts succeed. However, the discourse forms part of the context of each utterance and, hence, recognition of its overall structure is vital to the process by which the hearer interprets the speaker’s utterance. How this is done is not clear but linguists have progressed enough to model it though they have not reached any consensus. Here I discuss the highlights of relevant theories. In later sections I discuss how these theories are integrated

into this communication system.

Polanyi [51] describes the Linguistic Discourse Model (LDM). This hierarchical model has four main discourse units: sequences, expansions (e.g., of topic), joins (e.g., by *and*), and interruptions. Higher level discourse structures are the “goals and purposes which we try to accomplish through the use of language” [51, p. 627]. Communication involves complete Interactions (i.e., conversations) that involve the performance of Speech Events. The author defines a Speech Event as “the type of activity which the participants believe themselves to be engaged in” [51, p.629]. Polanyi points out that since the Speech Event involves the beliefs of the participants, misunderstandings can cause the participants to have different pictures of the current discourse structure.

Litman & Allen describe a discourse model that is plan based [38]. Their work builds on Cohen & Perrault’s work which generally requires that the actual conversation must closely follow the specified task structure [13]. Litman & Allen’s model allows a dialog to have unplanned subdialogs: topic changes, clarifications, and corrections. The authors also note that utterances can be about the plan itself. They can describe a step in the plan, change a plan, clarify a plan, or correct a plan [38, p. 164].

Higher level goals and activities justify the conversations we engage in. They also form part of the context and help the hearer understand the utterances he receives. Though this is not strictly part of a communication theory, it can help us understand (and, therefore, process) messages. Further, since discourse is a process, it would be convenient if we could apply our discourse model to more general processes or if the discourse model were managed within a more general process modeling system.

As mentioned above, planning forms the basis of some communication theory. A large literature on planning and reasoning about plans exists (e.g., [1, 39, 42, 47]). In these theories plans are essentially networks of activities with precedence defined. Both Petri nets [49] and statecharts [22] are well-studied formalisms for representing such graphs. Researchers have made many extensions to Petri nets (e.g., [6, 11, 35, 50]) that have made these graphs either more perspicuous or more expressive. Since Petri nets are powerful enough to model general processes, they are certainly sufficient for modeling conversations. That it can do it is non-controversial—Petri nets with inhibitor arcs are Turing machine equivalent [49, p. 201]. Many researchers have found Petri nets to be useful for representing business systems: Heuser & Richter [23], Oberweis & Lausen, [47], Richter & Voss [52], and Zisman [56]. Lee has used them for bureaucratic systems [34]. Statecharts have been used for many types of systems because they are compre-

hensible to people and allow them to construct and examine procedures; for example, Gordon and Moore use them to represent work processes, including those that required communication tasks [20]. No matter which language is used to describe conversational structure, terms representing this information can be integrated into the content vocabulary so that conversation partners could exchange information about conversations.

4 Proposed solution

Though some understanding of SAT is necessary, the purpose of this paper is not to investigate SAT deeply—it is to develop a communication model that can both express a wide variety of message types and interpret those messages with a flexible and modular system. In this section I define the language, the FLBC, and interpretation system, the MMS, that are based on the theory described in the previous section. This communication system explores and exploits the benefits of a linguistics-based automated electronic communication system and overcomes the shortcomings of previously defined systems. The impact of this system and the need for further research on it are described in the next, final, section.

4.1 The language

The basic structure of the formal language for business communication (FLBC) is given in Figure 3 and a portion of the Extended Markup Language Document Type Definition (XML DTD) is in Appendix A.

Here I discuss the language’s most distinguishing features with the aid of two examples. First, consider the message in Figure 4. The structure of the message is that every element is surrounded by two terms, one marking the element’s beginning (e.g., `<flbcMsg>`) and one marking the ending (e.g., `</flbcMsg>`). Additionally, in some beginning markers attributes can be listed along with the value assigned to it; e.g., `msgID` is an attribute of the `flbcMsg` element and is assigned the value of `msg12`.

Further inspection of the message reveals that `flbcMsg` is composed of only the `simpleAct` and `context` terms; the `content` term is a component of the `illocAct` term. Attributes of the `simpleAct` term specify that `nancy` is the sender, or `speaker`, of the message and `mackenzie` is the intended recipient, or `hearer`, of the message. The `simpleAct` term contains only the `content` term which, in turn, contains only the `predSt` term, `absent()`. The `force` attribute of `illocAct` specifies that the illocutionary force of this

```
<flbcMsg>
  <simpleAct speaker="speaker" hearer="hearer">
    <illocAct force="force">
      content
    </illocAct>
  </simpleAct>
  <context>context</context>
</flbcMsg>
```

where the slots are defined as follows:

speaker sender of this message

hearer recipient of this message

force illocutionary attitude expressed by the sender in this message

content propositional content

context context of the message; information that the speaker thinks would help the hearer properly understand the message

Figure 3: Basic structure of an FLBC message

```
<flbcMsg msgID="msg12">
  <simpleAct
    speaker="nancy" hearer="mackenzie">
    <illocAct force="predict">
      <predSt>absent(nancy, "08/07/1998",
        "08/08/1998")</predSt>
    </illocAct>
  </simpleAct>
  <context newConv="yes">
    <convStack stack="6" />
  </context>
</flbcMsg>
```

Figure 4: A simple FLBC message

message is `predict`. You should now see that this message embodies the F(P) framework; specifically, `predict` is the F and `absent()` is the P. The illocutionary force (the `force` attribute, or the F of the F(P) framework) is restricted to a small set of values—those illocutionary forces that are recognized by the system. The above portion of the message is translated as follows:

Nancy predicts to Mackenzie that Nancy will be absent from August 7, 1998 through August 8, 1998.

The XML DTD provides default values that are included in the message even though they are not explicitly represented. Specifically, it states that this message uses version 3 of the language definition (the definition given herein), uses the `norm` vocabulary, and that the message should be taken `literally`. `Literal` indicates to the recipient that the speaker is not trying to say anything more than what the message contains. The alternative to `literal` is `any`. This has been defined for completeness but is not currently integrated into the message interpretation system.

The `context` term specifies that this message begins a new conversation (`newConv="yes"`) and that this conversation should be identified by the identifier 6 (the value of the first, and only in this case, item in the list enclosed within the `convStack` element).

Most of the above will not be discussed again unless it affects either a particular message's interpretation or how the recipient would respond to it. The above discussion plus examination of Appendix A provides enough background to properly interpret the messages that follow.

The previous example (Figure 4) shows the basic structure of an FLBC message. A message can have a more complex structure, containing messages within messages. Consider the slightly more complex example shown in Figure 5. This message is translated as follows:

Lindsey requests that Nancy report to Lindsey whether or not it was true that Nancy was absent from August 7, 1998 to August 8, 1998.

Here is an instance of a recursively-defined message. Moore [43] found that messages in at least some formal languages that have been deployed (e.g., EDI and IAC) can be naturally expressed as embedded messages. In the FLBC, recursion (or embedding) can continue indefinitely, but in this case only one message is embedded. The message's structure is $F(F_1(P_1))$, where F is `request`, P is $F_1(P_1)$, F_1 is `report`, and P_1 is `isTrueFalse(absent(-nancy, "08/07/1998", "08/08/1998"))`.

```

<flbcMsg msgID="m72">
  <simpleAct speaker="lindsey" hearer="nancy">
    <illocAct force="request">
      <simpleAct speaker="nancy" hearer="lindsey">
        <illocAct force="report">
          <predSt>
            isTrueFalse(absent(nancy, "08/07/1998",
              "08/08/1998"))
          </predSt></illocAct></simpleAct>
        </illocAct></simpleAct>
      </illocAct></simpleAct>
    </illocAct></simpleAct>
  <context>
    <convStack stack="16" />
  </context>
</flbcMsg>

```

Figure 5: An FLBC message

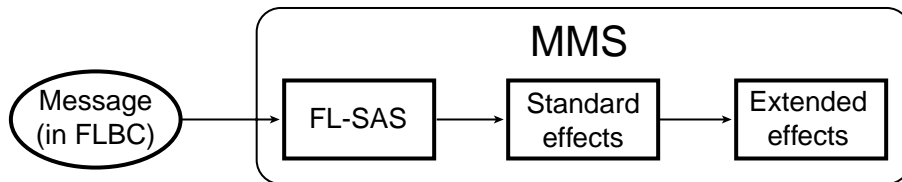


Figure 6: The structure of the message interpretation process

4.2 Responding to messages

The FLBC is expressive and flexible; however, to truly take advantage of this power, the process of responding to these messages must be equally flexible. The system for responding to messages is called a message management system (MMS). Figure 6 shows the overall structure of the message response process. The FL-SAS is the process message recipients go through in order to understand an FLBC message; unsurprisingly, it is closely analogous to the SAS for human communication. A message's `standardEffects` are modelled after the definitions of illocutionary forces discussed in §3.1 and defined in Appendix B; again, this is closely related to Bach & Harnish's work. Finally, a message's `extendedEffects` are those actions the hearer takes in response to the message which depend on both the message's propositional content and the context in which it was uttered. This goes beyond their work and incorporates research into discourse modeling discussed in §3.2.

In the following section I describe each of these steps in detail and show how each contributes to the process outlined in Figure 8.

4.2.1 Formal language speech act scenario

The purpose of the MMS is to interpret FLBC messages and respond to them appropriately. Bach & Harnish proposed that the SAS accurately models how humans communicate. Here I am proposing a model for automated electronic communication based on the SAS. I call this new model the *formal language speech act scenario* (FL-SAS, shown in Figure 7).

The FL-SAS is closely related to the SAS. The nine steps of the FL-SAS are analogous to the left column of the SAS as shown in Figure 1 while the four assumptions of each are nearly equivalent. The sequence of steps below shows the results of going through these nine steps for a specific message. The hearer (**Kenzie**) has received a message from the speaker (**Lindsey**). The speaker intends the hearer to reason as shown below. This presents the process of going through both the FL-SAS (Figure 7) and its implementation (Figure 8). The implementation is shown in Prolog even though a current version is being developed in Java. I use Prolog in this paper to describe the logic because I believe its syntax is more easily understood.

1. FL-SAS: Lindsey sent a message m . In this case m is equal to

```
<flbcMsg msgID="msg14"><simpleAct speaker=
"Lindsey" hearer="Kenzie"><illocAct force=
"question"><predSt>isTrueFalse(inStock(p12))
</predSt></illocAct></simpleAct><context
newConv="yes"><convStack stack="7"/>
</context></flbcMsg>
```

The message m is merely a string of bits with no assigned meaning. Think of the above simply as ASCII codes that have been converted to text for display purposes.

Implementation: This begins with the first step of the `flsas()` predicate, `hearsMsg(H, Msg)`. The variable `H` is set to the hearer's name when `flsas()` is called. `hearsMsg()` then sets `Msg` to the first string in the hearer's in-box.

2. FL-SAS: Lindsey means \mathcal{F} by m , where \mathcal{F} equals

```
<flbcMsg msgID="msg14">
  <simpleAct speaker="Lindsey"
```

In the following let $\mathcal{F} = \langle \text{flbcMsg} \rangle \langle \text{simpleAct speaker="S" hearer="H"} \rangle \langle \text{illocAct force="F"} \rangle \text{C} \langle / \text{illocAct} \rangle \langle / \text{simpleAct} \rangle - \langle \text{context} \rangle \text{X} \langle / \text{context} \rangle \langle / \text{flbcMsg} \rangle$. S intends H to reason as follows:

1. S sends a message m.
2. S means \mathcal{F} by m.
3. S is saying that $\ast(\dots \mathcal{F} \dots)$.
4. S is F-ing to H that C and considers X to be the minimal set of MCBs [mutual contextual beliefs; see below].
5. S could be F-ing to H that C considering X and other MCBs.
6. S is F-ing to H that C.
7. S could not be merely F-ing to H that C.
8. There is some F₁-ing that C₁ connected in a way identifiable under the circumstances to F-ing that C, such that in F-ing that C, S could also be F₁-ing that C₁.
9. S is F-ing that C and thereby F₁-ing that C₁.

Underlying assumptions:

Linguistic presumption The members of the community share a set of illocutionary forces, a message format, and a vocabulary. If a message is well-formed and valid, then the hearer can know or has the capability to come to know the meaning of the message.

Communicative presumption When a speaker sends a message to a hearer, the speaker has a recognizable intent for doing so.

Requirement of literalness When a speaker sends a message, he is required to speak literally.

Mutual contextual beliefs Mutual contextual beliefs that the speaker believes would help the hearer understand the message are contained within the **context** term of the message.

Figure 7: Formal language speech act scenario (FL-SAS)

```

mms(H) :-
    flsas(H, DirMsg, IndirMsg),
    perlocutionaryEffects(H, DirMsg,
IndirMsg).

flsas(H, DirMsg, IndirMsg) :-
    hearsMsg(H, Msg),
    parseMsg(Msg, ParsedMsg),
    spkrIsSaying(ParsedMsg,
IsSayingMsg),
    checkLocutionary(IsSayingMsg,
PossibleIlloc),
    evaluateDirect(PossibleIlloc,
DirMsg),
    evaluateIndirect(ParsedMsg, DirMsg,
IndirMsg).

hearsMsg(H, Msg)
    /* simply retrieves message from
in-box */

parseMsg(Msg, ParsedMsg)
    /* Translates from a text string to
a
    well-formed and valid XML flbcMsg
document
    (i.e., message) */

spkrIsSaying(ParsedMsg, IsSayingMsg)
    /* Identifies referents in the
message; ensures
    that hearer can identify them;
flags obvious
errors */

checkLocutionary(IsSayingMsg,
PossibleIlloc)
    /* determines a possible
illocutionary act from
    what is said and its context;
also verifies that
    the proposition is compatible
with the force */

evaluateDirect(PossibleIlloc,
DirMsg)
    /* Determines the unambiguous
meaning of the
illocutionary act; determines if
this meaning
is possible */

evaluateIndirect(ParsedMsg, DirMsg,
IndirMsg)
    /* Determines if an additional
illocutionary act
is possible; proposes the
additional meaning */

perlocutionaryEffects(H, Direct,
none) :-
    standardEffects(H, Direct),
    extendedEffects(H, Direct).

perlocutionaryEffects(H, Direct,
Indirect) :-
    standardEffects(H, Direct),
    standardEffects(H, Indirect),
    extendedEffects(H, Indirect).

```

Figure 8: Implementation of the FL-SAS plus the hearer's response to the utterance

```

        hearer="Kenzie">
    <illocAct force="question">
        <predSt>
            isTrueFalse(inStock(p12))
        </predSt>
    </illocAct>
</simpleAct>
<context newConv="yes">
    <convStack stack="7" />
</context>
</flbcMsg>

```

The bit-string *m* has now been interpreted and been properly recognized as an XML `flbcMsg` message. The utterance act has apparently succeeded.

Implementation: The `parseMsg(Msg, ParsedMsg)` term verifies that `Msg` term (the \mathcal{F} from above) is a valid and well-formed XML `flbcMsg` message. At this point `parseMsg()` also converts the message into a form amenable to internal processing. (For Java this would be some type of object.)

3. FL-SAS: *Kenzie* checks that she knows what a `p12` is and what `inStock()` and `isTrueFalse()` mean. All are known. *Lindsey* is saying that she wants *Kenzie* to tell her if `p12` is in stock. The locutionary act has apparently succeeded.

Implementation: In `spkrIsSaying(ParsedMsg, IsSayingMsg)` the system verifies that each of the terms within the XML `flbcMsg` message is a known term. In this case it looks up `p12`, `inStock()`, and `isTrueFalse()` in a dictionary.

4. FL-SAS: *Lindsey* is questioning *Kenzie* if `p12` is in stock. Further, this message (`m14`) begins a new conversation (`newConv`).

Implementation: The `checkLocutionary(IsSayingMsg, PossibleIlloc)` term considers all parts of the message and determines a possible illocutionary act. It also verifies that the propositional content is locutionary compatible with the force of the message. Predicates that define this type of compatibility are shown in Appendix C. This example is compatible because a `question` requires that the `content` be structured as `isTrueFalse()`.

5. FL-SAS: *Lindsey* is a past customer of *Kenzie*'s. The item `p12` is an

item for sale to the general public. It is feasible that *Lindsey* could be questioning *Kenzie* if *p12* is in stock.

Implementation: The `evaluateDirect()` predicate evaluates the message within its context to determine if it seems feasible. The system verifies that it not only knows what the terms in the message mean—this was verified in step 3—but that this particular composition of terms makes sense.

6. FL-SAS: *Lindsey* is questioning *Kenzie* if *p12* is in stock. The illocutionary act has apparently succeeded.

Implementation: This is simply the output of the `flsas()` predicate, where `DirectMsg` ends up containing the illocutionary act that *Kenzie* thinks *Lindsey* performed.

Kenzie stopped at step #6 because she was satisfied that *Lindsey* was simply questioning her. As per the FL-SAS's "requirement of literalness" (compare with the "presumption of literalness" in the SAS) the hearer's inferences *must* be based on the literal meaning of the message. This simplifies the MMS, allowing the FL-SAS to include only those steps related to literal messages—without unduly limiting the communicative system for now.

This compromise highlights an important aspect of this system. It is always possible to describe a system which is more complex, or more realistic, or more general—but it is not always desirable. In this case, a more flexible method of message interpretation would allow techniques such as non-literal speaking. This may be the most flexible but it also makes interpretation and message formation difficult. For example: If the sender of a message does not know how the message is going to be interpreted, then how will it know what kind of message is necessary to get the recipient to act in the desired manner? With this in mind it is clear that adding the ability to speak non-literally should be done carefully. On the other hand, leaving this capability out will limit the potential usefulness of the language. I have taken a position in the middle. Design the system based on a theory which provides a blueprint for expansion but implement a simplified version which may provide most of the benefit without much of the difficulty.

The FL-SAS's other assumptions are similarly related to those used in the SAS. The "linguistic presumption" simply means that there is a community who can communicate among themselves. The "communicative presumption" means that hearers can assume that speakers will not send out random messages or meaningless messages; rather, the hearer can assume that the

```

standardEffects
  flbcMsg(speaker=S, hearer=H, force=assert,
          content=C, context=X, id=ID) :-
considerForKB S believes C
                because flbcMsgID=ID,
considerForKB S wants (H believes C)
                because flbcMsgID=ID.

```

Figure 9: Standard effects for an assertion

speaker is sending a message for a reason and his intent will be obvious. Finally, the “mutual contextual beliefs” assumption requires that the speaker include information in the `context` portion of the message if the speaker believes that the information would help the hearer interpret the message.

Returning back to the example, Kenzie has now inferred that Lindsey is questioning her. She has (presumably) successfully completed the FL-SAS—she understands what Lindsey’s intentions were in sending her this message. But that is all that has happened. Kenzie has not responded to the message. Kenzie has not even necessarily decided that she is going to respond to this message. All she has done is gained an understanding of Lindsey’s intentions in sending the message. According to the series of four acts which Austin proposed, this speech act has successfully completed the utterance, locutionary, and illocutionary acts but has yet to complete the perlocutionary act. My interpretation of what should occur in this act is discussed in the following two sections. Basically, I propose that this act be broken into two pieces. Now on to the details.

4.2.2 Standard effects

The standard perlocutionary effects, which are also referred to as the *standard effects* of a message, are those that occur as a result of the success of the illocutionary act. Consider again the definition of *assert* in Figure 2 and the discussion of it in §3.1. The problem currently in front of us is how to automate this process. The definition of the `standardEffect` for an *assert* message is given in Figure 9 and in Appendix D (using Prolog with several operators defined to increase legibility) along with definitions for all the other illocutionary forces. The interpretation of this is as follows:

If the hearer `H` receives a message from `S` that is an **assertion** that `C`, then the hearer should consider adding to his knowledge base that `S` believes that `C` (based on his assertion). `H` should

also consider believing that **S** wants **H** to believe that **C** (again, based on this assertion).

According to this definition the hearer is not required to believe—to add a belief to his knowledge base—that the speaker believes that **C** because of the speaker’s assertion. He is only required to consider believing it. However, given that there is a presumption of sincerity, the implementation of the `considerForKB` predicate is probably fairly simple: if someone tells you something about their beliefs or desires, then believe what they say. In the case of this assertion, the hearer would believe that **S** believes **C** and that **S** wants **H** to believe that **C**. The hearer *does not* have to believe that **C**, he only has to believe that the speaker believes it and that the speaker wants him to believe it. It is then an entirely new process for the hearer to evaluate how his own beliefs will change in response to his new belief that the speaker wants him to believe **C**. I call this the “anti-brainwashing principle.”²

Requiring that the hearer believe what the speaker wants him to believe would be a disaster for electronic commerce. Consider the following scenario:

An unfamiliar company sends your company an assertion that their computer is superior to all other computers. You add this to your knowledge base since the message handler requires it. All future computer purchase decisions then take into account that this company’s quality is higher.

This may seem like a fallacious comparison since companies would not implement this message handler in this way. However, a standard implementation strategy would be to add a special message handler for this message and have it determine if the company is known, if it is trusted, etc. This would not be difficult but it leads the implementor down a path that requires that almost every message (hundreds and thousands of them) have a special handler to take into account the vagaries of each message. This is not what we want to do. What we want is to have as few message handlers as possible and limit the places where these handlers might need to be changed. This is what the FL-SAS provides us. If it is correct, the FL-SAS will not need to be changed, the standard effects will not need to be changed, a few standard effects might, but should not, have to be added, and the rest of the changes will be located in the `extendedEffects` and `considerForKB` predicates. This could be a tremendous breakthrough, allowing much easier construction of extensive communication-based applications.

²Thanks to Michael Covington for suggesting this term.

In order to implement a system that takes advantage of all that the FLBC and MMS combination has to offer, a powerful and flexible knowledge base management system would have to be in place. As Lee [34] and Kimbrough & Moore have pointed out [28], the system would, at a minimum, have to be able to reason defeasibly about obligation and time. This is a difficult problem in itself but Kimbrough & Hua have proposed a promising method [26].

4.2.3 Extended effects

While standard effects are content independent, extended effects are dependent on both the message, its force and content and context, and its effects on the recipient of the message. Extended effects are the perlocutionary effects of a message which cannot be predicted based solely on the message's illocutionary force and which cannot be specified by any theory beforehand. All that can be done here is to specify the general form of these effects and how they might be specified.

Certainly it is true that the questions "What time is it?" and "Where is the nearest fire station?" have something in common: they are requests for information and, as such, the hearers would recognize, if the communicative acts were successful, that the speakers want the hearers to provide some information. However, it is also true that these requests differ at some basic level and provoke different reactions by the hearers. The second question would probably provoke different combinations of activity and follow-on questions while the first would probably necessitate no further action or interaction. The MMS captures the similarity in its standard effects and the differences in its extended effects.

Extended effects are rules about conversations (to be begun or continued or interrupted) or processes (to be begun or continued or interrupted) which get invoked as a result of a message being received and the hearer changing his beliefs or goals. Consider again the message in Figure 5. This is a simple request by Lindsey for information from Nancy. Nancy's MMS handles this message in the following manner. The FL-SAS receives the message and eventually recognizes that it is a request for information. The `standardEffects` of this message are 1) that Nancy consider whether or not she believes that Lindsey wants Nancy to report some fact to Lindsey, and 2) that Nancy consider whether or not she desires to do so. If Nancy thinks Lindsey is sincere (the default assumption), then the result of this message is that Nancy believes both that Lindsey wants Nancy to report some fact and that Lindsey wants Nancy to desire to report that fact. The

1. **Command based**

- If I get a request, do what is asked.

2. **Limited-command based**

- If I get a request from Lindsey, do what is asked.
- If I get a request from a person who belongs to a pre-specified group of people (e.g., those whom Nancy considers trustworthy), do what is asked.
- If I get a request from a person who belongs to a pre-specified group of people, send a reply stating that I do not have the time to fulfill the request.

3. **Self-knowledge based**

- If I get a request and I want to do what is asked (that is, the requested action has been added to Nancy's goal list, probably as a result of the standard effects of this request), then do what is asked.

4. **Other-knowledge based**

- If I get a request and I want to do what it asked and I do not believe that my boss believes that I should not do this, then do what is asked.

5. **Conversational-structure based**

- If I get a request and I want to do what it asked but I do not know the purpose of the conversation, then I interrupt the conversation to determine its purpose.
- If I get a request to do something which I need more information about, then I will delay continuing this conversation and start a conversation with another party in order to get the needed information.

Figure 10: Sample rules for responding to a *request*

`extendedEffects` which are applicable to this message can have several forms, and several levels of generality.

As an aid to exploring these forms, consider the rules which Nancy might have for responding to a request (Figure 10). These extended effects determine the complexity of different MMSs. An FLBC and MMS based communication system is complex, requiring numerous inter-connected parts to be developed simultaneously. It would be unrealistic to think that a company could initially deploy a complete system integrating complex indirect speaking and reasoning about the beliefs of others. Some manner of growing the system must be considered. These methods of responding to a message,

taken together, allow for increasing sophistication while either integrating this communication system into a company's operations or experimenting with its capabilities. This list is not complete by any means but is meant to be suggestive of a plan of implementation. The appropriate level of implementation depends on the skill level of the developer, the requirements of the application, and the developer's familiarity with the FLBC and MMS.

The last system in this list needs some explaining. The *conversational-structure based* system adds the ability to respond to a message based on what the system *expected* the message to be, given its own model of how the conversation should progress. (More on this is provided in Moore [45].) This addition requires that the system maintain in its knowledge base a set of planned conversation structures. The system can use these as a base of comparison for the flow of the current conversation. These structures could be represented externally as statecharts (though, of course, internally they would be represented by some equivalent data structure). If the hearer determines that the incoming message, which is part of an on-going conversation, requires further clarification, then the hearer can start a sub-conversation as part of its strategy in formulating a response.

This need to manage conversations is the reason for the predicates in the `<context>` term (shown in Appendix A): `respondingTo`, `convStack`, `interruption`, `subordination`, and `correction`. It is standard practice in creating a communication system to include information about the message that the current message is a response to; simple examples are the practice of "quoting" the original message when replying to an email message and the practice of putting `Re: X` in response to a message whose subject was `X`. In the FLBC this simple response information is indicated by the inclusion of the `respondingTo` term. Somewhat redundantly, but at little cost, the `convStack` term indicates which conversation the current message is part of. If it is a new conversation, then the `newConv` attribute of the `context` term is set to `yes`.

What is somewhat unique, and very useful, about this system is that actual discourse is allowed to deviate from the predefined conversation plan (for more details on this, see [45]). This is done by incorporating terms in the `context` that signal to the recipient that the current message is related in some particular way to other messages. Each term implicitly tells the conversation partner: "Wait, there's something else I want to or need to talk about before we proceed with the current conversation." Four terms are necessary to model subdialogs that researchers have identified (see §3.2). `RespondingTo` is discussed above. This corresponds to Polanyi's coordination type. The `interruption` term indicates that a new dialog should

temporarily become the focus. When interrupting a conversation (or when using either of the following two terms), a new conversation identifier is also added at the beginning of the `convStack` term. Conversations can be nested arbitrarily deep. A `subordination` term in the `context` indicates that the message is elaborating on a point made in a previous message. This message does not have to match one of the messages that are expected to occur next in the conversation. It should be about the previous message, probably as a clarification of some fact. `Correction` indicates that the message somehow corrects a previous message in the conversation.

Labrou & Finin have defined conversation policies for the KQML performatives [32]. They use these conversation policies to restrict possible sequences of messages much more than the way proposed in this article. According to their proposal, if an incoming message does not fit in an ongoing conversation and cannot be used to start a conversation, then it is considered to be in error [32, p. 453]; thus, the incoming message would have had no effect at all on the recipient. The current proposal is not so harsh. After a message's standard effects are realized, then the system attempts to match it against ongoing conversations. If no match is made, then the system starts a new conversation by default. Since conversations can be constructed dynamically, almost any message can follow any other message. The current proposal does not use conversation policies to restrict message flow; rather, it is used to provide *ex ante* guidance as to how messages *usually* flow and *ex post* declarations of which messages are related to each other. It is up to further experimentation to determine which of these provides more practical benefits and is more easily implemented.

Discourse modeling can become a fruitless exercise in complexity management if some method for breaking into subdialogues is not provided. Otherwise, countless contingencies and contingencies for these contingencies have to be planned for. Again, the idea here is to modularize. Plan what a normal conversation would look like. Build other plans for handling clarifications and corrections—these can be called *as needed* at any point in the conversation, not just where a planner might have foreseen that it was necessary.

5 Conclusion

The FLBC language and MMS exploit linguistics throughout their design: The FLBC is based on the F(P) structure. Messages are allowed to take a wide range of illocutionary forces. Messages are defined by the speaker's in-

tended effects. The actual effects (e.g., perlocutionary effects) of a message are determined by the recipient and not by the sender—just as the actual effects of a natural language message are determined by the hearer and not by the speaker. Each message’s standard effects are determined through recourse to the definition of their natural language counterparts. Messages are interpreted within a speech act theory inspired procedure (e.g., the FL-SAS). Natural language discourse modeling conventions are integrated into conversational modeling for FLBC messages. It is this depth of integration, this usage of a wide range of influencing linguistic concepts, that is the real contribution of this system. While not wholly novel, this system’s strong reliance on linguistics is unique. The motivation for working with the complex theory is to see if the greater start-up costs will pay off with decreased effort as more messages are implemented.

Assumptions underlying this research that have so far gone unstated are that this whole approach is appropriate for automated electronic communication and, more to the point, both that dividing the message interpretation process into pieces as the FL-SAS does is beneficial and that linguistic interpretation of messages is useful. There are two senses of disbelief that might rear themselves in someone encountering this theory. First, it might be believed that the FL-SAS does not describe anything more involved than what is done with normal computer-based message interpretation. It is merely the unfamiliar terminology that makes this theory appear more complex than it really is. Second, it might also be believed that the FL-SAS does not add any value to the “standard” message interpretation process but simply makes it more complex. The message ends up being interpreted just as it would without the added complexity and the response is essentially the same. A simple answer to the first point is that it would be surprising if processing a computer-based message were *that* much different than a basic description of natural language processing. It should raise our confidence instead of our doubts to see the similarity between what is normally done and what this theory states. In the following I provide a more complete answer to the first point and an argument against the second.

Though there are similarities between what is normally done and what this theory describes, significant differences do exist and these make a difference in the message interpretation process. Certainly, step #1 (`hearsMsg()`) of the FL-SAS is nothing surprising. Bits are received—in natural language this act is simply the person hearing the sounds of the utterance. Step #2 (`parseMsg()`) determines that the message is well-formed and valid. Again, this is not particularly surprising; a reasonable first step after receiving a message is to determine if the message is constructed properly. As an aside,

one benefit of using XML is that this validation process is automated with commonly-available tools. The message is now meaningfully processable—we know that the bits are ordered in an appropriate way and that we can justifiably begin to ask what it means and how we should respond to it. It is at this point in the process, before the meaning of the message is even explored, that I propose that “normal computer-based message interpretation” begins to formulate a response to the incoming message. Consider this fictitious, though plausible, alternative (call it the *standard* system) in which an EDI-like message is received representing the same content as used in an earlier example:

Some bits are received: "***14*yes*7*question*Lindsey*Kenzie*isTrueFalse*1*inStock*1*p12***". The system checks that the message has the correct number of fields; it does. This message has id 14 and begins a new conversation (**yes** & 7). This is a **question** message. The system verifies that it knows how to handle this type of message. The next two fields reveal that this message is from **Lindsey** to **Kenzie**. The **content** term begins with the term **isTrueFalse**. The first 1 tells the system that **isTrueFalse** has one argument. [It continues through the rest of the fields.] The system now invokes the message responding procedure (or, maybe, the **question** responding procedure) with all the above arguments. If this procedure call succeeds, then a response is formulated; if it fails, then a message is sent to the originator of the message saying that **Response could not be formulated. Please try again** (in either plain text or automated form—it does not matter).

After the system verifies that the message has the correct number of fields (in the second sentence above), then, in the terminology of SAT, there has been a successful utterance act. This is where the standard system begins to formulate a response. The system does not try to determine the meaning of the terms. It simply gathers up arguments to be passed to another procedure. Responding to a message in this manner is analogous to Pavlov's dog's response to a ringing bell. The ringing does not *mean* anything, it simply signifies something. The dog's learned responses are limited to the number of signals that it can learn, and there is a one-to-one correspondence between the signal and the response. For the communication system, just as with the dog, use of a message must be foreseen so that appropriate response mechanisms can be created. While sufficient for simple communication systems (bell, dog, food), I propose that it is obvious that a signal-based

communication system will not be able to support a large number of complex signals—either the whole system will become too complex to maintain because of add-ons that have to be created to handle unforeseen changes to the system or the signals will get too complex to interpret correctly.

On the other hand, the FL-SAS and the MMS specify that message responses rely on inferences drawn from the message’s meaning, and not simply from its form. Since the utterance act does not delve into the meaning of the message, it does not provide an appropriate basis for interpreting messages that are part of a complex messaging system. In addition, the FLBC allows messages to be constructed by assembling existing terms in new ways. This allows for the creation and exchange of many messages that were not planned for. All of this flexibility has many advantages. Planning the system can be less thorough and implementation more iterative since messages can be added more easily to the system. Defining a few terms implicitly defines many more messages because of the ability to combine terms in multiple ways. I contend that these advantages all apply to electronic commerce.

The process of performing the locutionary and illocutionary acts result in the receiving system checking a message for errors of a certain type and in a certain order. This is in contrast to the standard system in which all the error handling—after initial format checking—was handled by the message responding procedure. What level of error handling, and in what order, is unclear in that system. Part of what the FL-SAS provides is a specification of what errors to check for and the order in which they should be checked. This should make error messages more meaningful and establish a minimal acceptable level of error checking. Further, the examples shown do not demonstrate, though it is clearly possible given the system’s ability to handle conversation interruptions (see §4.2.3), that the system provides some support for automated correction of error states. For example, if during execution of the locutionary act Kenzie’s system determines that it does not know what one of the symbols refers to, then it can interrupt the conversation, and its own interpretation of the message, and ask Lindsey’s system to clarify the reference. After receiving the clarification, Kenzie’s system can attempt to perform the locutionary act again (and proceed on to the rest of the FL-SAS). This same type of error correction mechanism could be used during the illocutionary act (i.e., `evaluateDirect()`) if the system determines that it is not able to handle a particular composition of terms even if the terms themselves are known.

The most significant distinction is made between illocutionary and perlocutionary acts. After completing the illocutionary act, the hearer presumably knows what the speaker’s intentions were in sending the message. Its

perlocutionary effects are those effects that are an intentional result of the success of the illocutionary act. Because this distinction between acts is made, the message's sender cannot know either the message's effects on the hearer or the responses the hearer might make. It is up to the hearer to decide what the effects of the message will be. KQML, for one, puts only the slightest barrier between the receipt of the message and having its effects take hold—the message receiver merely has to have asked for this information earlier. Having received a request for this information, a sender *knows* what the effects of a message will be when sent to the agent who sent the request. Certainly separating the message interpretation from its actual effects is not always appropriate—think of a control application in a nuclear power plant. When a control message is sent, the operator wants to be able to predict with perfect accuracy what the effects of the message will be. I propose that this separation *is* appropriate for electronic commerce with many participants and many message types or for any application whose automated communication needs are going to evolve.

Given the above explanation, it appears that the FL-SAS does differ from standard practice and that it does add value to the message interpretation process. The advantages and implications described appear to support the contention that the FL-SAS could be beneficial to automated electronic communication.

5.1 Impact

Even if the FLBC and MMS are not going to replace EDI anytime soon, they can still be used as add-ons to existing EDI standards. Many EDI standards (if not all) have free-text messages. This message could carry an FLBC message. When the system receives a free-text message, in a pre-processing routine it could check for the existence of an FLBC message. If one is detected, then it could be passed to the MMS for processing. Hurst *et al.* demonstrated this to a limited extent with an existing, proprietary EDI system [24].

The proposed communication system promises benefits for application developers and users. Once the basic communication infrastructure is in place, it should be easier to add new communication based features to programs. Its architecture should make it easier and less time-consuming to deploy new applications and modify existing ones. This modularity would help programmers develop more consistent and more easily understood message definitions. Reusability and modularity are clear advantages when there is a programmer shortage as there is today and will be for the next ten years

[5]. This solution could be used to support inter- and intra-organizational communication in its many guises: EDI, workflow processing, active (or computational) mail [10], agent communication language (e.g., KQML [16]), supply chain management by agents [8], inter-application communication (IAC [2, 43]), Web-based commerce, and electronic mail. As more programs gain these features, programmers will have more opportunities for exploiting their capabilities. Finally, individual applications will have more sophisticated communication facilities than they would have without this system because many people in many companies would be contributing to the growth of the system. Moore & Kimbrough [46] demonstrated many of these benefits with two prototype application built on an earlier version of this communication system.

On another practical front, I defined the FLBC in the XML language because it is a language-independent data-interchange format. Since XML-parsing tools and libraries should be soon appearing in most programming languages (and have already in Java, Perl, and Tcl), it should be easy to develop the tools to compose, send, receive, and read these messages in whatever programming language is desired. Further, it should soon be easy to create Web pages—with commercially-available Web page creation tools—that include a form that can accept data and send a valid XML message. This could speed the acceptance of the FLBC and provide a simple way to experiment with the language.

5.2 Future research

I take seriously several hypotheses. First, SAT provides a framework for understanding utterances of all types. Bach & Harnish call their interpretation of this procedure the SAS. If SAT is correct, then this general procedure can be used to understand *any* utterance. While it might be interesting to attempt to disprove this assertion, a more fruitful investigation would be to determine how useful this procedure is compared to competing theories.

Second, and more to the point for the current investigation, the FL-SAS provides an approximation of the SAS that is appropriate for electronic messages and automated handling of those messages. If this second hypothesis turns out to be true, then several benefits must immediately follow. First, the FL-SAS's structure will not have to be changed no matter how many message types are defined. Second, the standard perlocutionary effects will not have to be changed no matter how many message types are defined. Third, only infrequently will additional illocutionary forces—and, hence, additional standard effects—need to be added to the list of implemented forces. Fi-

nally, all changes necessary to respond to a message will be located in the extended effects. The appropriateness of the FL-SAS approximation would be difficult to test directly. What could be tested is the existence of the benefits. If these do hold, then it could be said that the FL-SAS is quite appropriate for electronic messages and the automated handling of those messages.

I also propose that, just as human communication skills get more sophisticated over time, electronic communication skills can do the same. Specifically, the inferential tools needed to support an EDI system based on this communication model can be made more sophisticated in steps, depending on the demands of the application using it. In §4.2.3 I proposed five methods of responding to messages that incorporate varying levels of complexity. Research should be undertaken to determine which of these methods are more appropriate, or necessary, for which types of organizational communication.

One problem common to all communication systems is that relating to the *extension* of each symbol: how to reliably and accurately link the symbol to the set of all real world objects which it is meant to represent. For example, it might be that the symbol *gw* has the extension the person who was the first President of the United States. All the work that researchers go through to ensure that the symbolic meaning of messages are correctly determined will go for naught if this extension problem is not solved or at least productively addressed. We send messages to accomplish tasks in the real world and it is through a symbol's extension that a message's meaning gathers import. I do not address this problem in this paper because of its scope but, separately, in another [30].

In order to incorporate the more advanced types of communication, a sophisticated knowledge management system must be developed. Ideally, this system should be able to reason defeasibly about obligation, belief, and time. Further investigation must determine just how much the communication model depends on the existence of such a system.

I have proposed that a message's standard effects provide a certain level of usefulness, independent of whether or not any extended effects are applicable to that message. Experiments must be performed to verify this proposition.

Management of the vocabulary for the content terms provides a fertile research area. It is entirely possible that some intelligence may have to be built into the vocabulary terms or that they may have to have some particular structure in order for the FLBC and MMS to provide their promised benefits. The extent of this intelligence or the nature of this structure will emerge through further theoretical development combined with experience

gathered during research.

One of this pair’s distinctive features is that it has the potential for providing a means of “bootstrapping” a system of automated communication. Since the system has a certain level of communication skills built in, and since terms that can be used to describe a conversation model and other extended effects are part of the vocabulary, it is possible that a system could be “told”—through a series of FLBC messages—how to respond to messages. First, the effectiveness and usefulness of this type of “conversational” interface for systems development should be investigated. If it passes these tests, then the requirements of such a system should be investigated and spelled out in more detail.

Castells has pointed out that in the informational economy (i.e., what has come upon us at the end of the industrial economy) “the source of productivity lies in the technology of knowledge generation, information processing, and symbol communication” [12, p. 17]. By supplanting or simply by augmenting EDI and other forms of inter- and intra-organizational communication, the FLBC and languages like it provide another opportunity for organizations to increase their productivity as the informational economy continues to spread.

A Definition of FLBC as XML

The following is a portion of the XML DTD for an FLBC message. The full DTD is located at <http://www-personal.umich.edu/~samoore/research/flbcMsg.dtd>. XML is a new International Standard (ISO 8879:1996) based on SGML which is being proposed as a general purpose, platform-independent, extremely flexible markup language. Light’s book [37] provides a readable introduction to XML. Two useful online resources are the World Wide Web Consortium (W3C) site at <http://www.w3.org/pub/WWW/XML/> and Robin Cover’s site at <http://www.sil.org/sgml/xml.html>. XML has the additional benefit of looking like HTML (the ubiquitous language used to construct most of the Web’s pages) and, thus, being familiar to and easy to read for many people. Previous versions of FLBC have been defined as either Prolog- or LISP-forms. This conversion to XML changes to a more conventional format but does not make any meaningful changes to the language.

A.1 Definition

```
<!ELEMENT flbcMsg (simpleAct, context) >
```

```

<!ATTLIST flbcMsg
  msgID ID #REQUIRED
  grammar NMTOKEN "4"
  vocab NMTOKEN "norm"
  literal ( literal | any ) "literal" >

<!ENTITY % Statechart.Elements " ... " >

<!ENTITY % Vocabulary "%Statechart.Elements; " >
<!ENTITY % Logical.Forms " andMsg | orMsg | isNotMsg | iffMsg |
  ifThenMsg | ifThenElseMsg " >
<!ENTITY % Simple.Content "%Vocabulary; | predSt |
  %Logical.Forms; " >
<!ENTITY % Content "flbcMsg | simpleAct | %Simple.Content; " >

<!ELEMENT simpleAct (illocAct+) >
<!ATTLIST simpleAct
  speaker NMTOKEN #REQUIRED
  hearer NMTOKEN #REQUIRED >
<!ELEMENT illocAct (%Content;)>
<!ATTLIST illocAct
  force ( advise | ascribe | assent | assert | concede | confirm |
    denial | describe | dispute | dissent | inform | offer |
    permit | predict | prohibit | promise | question | request |
    require | retract | report ) #REQUIRED>

<!ELEMENT predSt (#PCDATA)>
<!ATTLIST predSt
  language ( prolog | lisp | kif | xml ) "prolog">
<!ELEMENT andMsg (%Content;)+>
<!ELEMENT orMsg (%Simple.Content;)+>
<!ELEMENT isNotMsg (%Simple.Content;)>
<!-- Interpreted as "It is not true that X" -->
<!ELEMENT iffMsg ((%Simple.Content;), (%Simple.Content;))>
<!-- Interpreted as "X is true if and only if Y is true" -->
<!ELEMENT ifThenMsg ((%Simple.Content;), (%Content;)) >
<!-- Interpreted as "If X is true, then Y is true" -->
<!ELEMENT ifThenElseMsg ((%Simple.Content;), (%Content;),
  (%Content;)) >

<!ELEMENT context ((timeSent | alsoSentTo | alsoAddressedTo |
  forwardedBy )*) >
<!ATTLIST context
  newConv ( yes | no ) "no"
  respondingTo NMTOKEN #IMPLIED

```

```

    sendingMachine NMTOKEN #IMPLIED
    convStack NMTOKENS #IMPLIED
    interruptType (interruption | subordination | correction )
        #IMPLIED >
<!ELEMENT timeSent (time) >
<!ELEMENT alsoSentTo (person+)>
<!ELEMENT alsoAddressedTo (person+) >
<!ELEMENT forwardedBy (person+) >
<!ELEMENT time EMPTY>
<!ATTLIST time
    year NMTOKEN #REQUIRED
    month NMTOKEN #REQUIRED
    dayOfMonth NMTOKEN #REQUIRED
    hour NMTOKEN #REQUIRED
    min NMTOKEN #REQUIRED
    sec NMTOKEN #REQUIRED >

```

A.2 A sample message

The message below shows what an FLBC message looks like when structured as a *well-formed* and *valid* XML message. I checked these messages for well-formedness and validity with the *DataChannel XML Parser* which I downloaded from <http://www.datachannel.com/products/xml/DXP>. The messages in the body of this paper were formatted correctly except for omitting the XML and DOCTYPE elements from the beginning of the message.

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE flbcMsg SYSTEM
    "http://www-personal.umich.edu/~samoore/research/flbcMsg.dtd">
<flbcMsg msgID="ID">
    <simpleAct speaker="lindsey" hearer="nancy">
        <illocAct force="request">
            <simpleAct speaker="nancy" hearer="lindsey">
                <illocAct force="report">
                    <predSt>
                        isTrueFalse(absent(nancy, "08/07/1998", "08/08/1998"))
                    </predSt>
                </illocAct></simpleAct></illocAct></simpleAct>
            </context></context>
        </flbcMsg>

```

As shown in the DOCTYPE element above, the DTD for an flbcMsg is located on the Web at the University of Michigan. I will leave this at the indicated address for the foreseeable future.

The above DTD allows the message to leave out some information where default values are commonly used. The valid and well-formed message returned by the parser—that is, with all default values represented explicitly—is the following:

```
<flbcMsg msgID="ID" grammar="4" literal="literal" vocab="norm">
  <simpleAct speaker="lindsey" hearer="nancy">
    <illocAct force="request">
      <simpleAct speaker="nancy" hearer="lindsey">
        <illocAct force="report">
          <predSt language="xml">
            isTrueFalse(absent(nancy, "08/07/1998", "08/08/1998"))
          </predSt>
        </illocAct></simpleAct></illocAct></simpleAct>
      </context newConv="no"></context>
    </flbcMsg>
```

This added information specifies the form of the grammar (**grammar**), whether the message is meant to be interpreted literally (**literal**), the vocabulary used (**vocab**), the form of the information in the context (**language**), and whether or not this message begins a new conversation (**newConv**).

B Definition of illocutionary forces

The following are definitions of the illocutionary forces classified as *constatives*, *directives*, or *commissives* on pages 42–55 of [4].

advisory (admonish, advise, caution, counsel, propose, recommend, suggest, urge, warn) In uttering *e*, *S* advises *H* to *A* if *S* expresses:

1. the belief that there is (sufficient) reason for *H* to *A*, and
2. the intention that *H* take *S*'s belief as (sufficient) reason for him to *A*.

ascriptive (ascribe, attribute, predicate) In uttering *e*, *S* ascribes *F* to *o* if *S* expresses:

1. the belief that *F* applies to *o*, and
2. the intention that *H* believe that *F* applies to *o*.

assentive (accept, agree, assent, concur) In uttering *e*, *S* assents to the claim that *P* if *S* expresses:

1. the belief that P, as claimed by *H* (or as otherwise under discussion), and
2. the intention (perhaps already fulfilled) that *H* believe that P.

assertive (affirm, allege, assert, aver, avow, claim, declare, deny (assert ... not), indicate, maintain, propound, say, state, submit) In uttering *e*, *S* asserts that C if *S* expresses:

1. the belief that C, and
2. the intention that *H* believe that C.

concessive (acknowledge, admit, agree, allow, assent, concede, concur, confess, grant, own) In uttering *e*, *S* concedes that P if *S* expresses:

1. the belief that P, contrary to what he would like to believe or contrary to what he previously believed or avowed, and
2. the intention that *H* believe that P.

confirmative (appraise, assess, bear witness, certify, conclude, confirm, corroborate, diagnose, find, judge, substantiate, testify, validate, verify, vouch for) In uttering *e*, *S* confirms (the claim) that C if *S* expresses:

1. the belief that C, based on some truth-seeking procedure, and
2. the intention that *H* believe that C because *S* has some support for *P*.

deny (assert ... not) In uttering *e*, *S* denies that C if *S* expresses:

1. the belief that not C, and
2. the intention that *H* believe that not C.

descriptive (appraise, assess, call, categorize, characterize, classify, date, describe, diagnose, evaluate, grade, identify, portray, rank) In uttering *e*, *S* describes *o* as F if *S* expresses:

1. the belief that *o* is F, and
2. the intention that *H* believe that *o* is F.

disputative (demur, dispute, object, protest, question) In uttering *e*, *S* disputes the claim that P if *S* expresses:

1. the belief that there is reason not to believe that P, contrary to what was claimed by *H* (or was otherwise under discussion), and
2. the intention that *H* believe that there is reason not to believe that P.

dissentive (differ, disagree, dissent, reject) In uttering *e*, *S* dissents from the claim that P if *S* expresses:

1. the disbelief that P, contrary to what was claimed by *H* (or was otherwise under discussion), and
2. the intention that *H* disbelieve that P.

informative (advise, announce, apprise, disclose, inform, insist, notify, point out, report, reveal, tell, testify) In uttering *e*, *S* informs *H* that C if *S* expresses:

1. the belief that C, and
2. the intention that *H* form the belief that C.

offer (offer, propose; also volunteer, bid) In uttering *e*, *S* offers A to *H* if *S* expresses:

1. the belief that *S*'s utterance obligates him to A on condition that *H* indicates he wants *S* to A,
2. the intention to A on condition that *H* indicates he wants *S* to A, and
3. the intention that *H* believe that *S*'s utterance obligates *S* to A and that *S* intends to A, on condition that *H* indicates he wants *S* to A.

permissive (agree to, allow, authorize, bless, consent to, dismiss, excuse, exempt, forgive, grant, license, pardon, release, sanction) In uttering *e*, *S* permits *H* to A if *S* expresses:

1. the belief that his utterance, in virtue of his authority over *H*, entitles *H* to A, and
2. the intention that *H* believe that *S*'s utterance entitles him to A.

predictive (forecast, predict, prophesy) In uttering *e*, *S* predicts that C if *S* expresses:

1. the belief that it will be the case that C, and

2. the intention that *H* believe that it will be the case that *C*.

prohibitive (enjoin, forbid, prohibit, proscribe, restrict) In uttering *e*, *S* prohibits *H* from *A*-ing if *S* expresses:

1. the belief that his utterance, in virtue of his authority over *H*, constitutes sufficient reason for *H* not to *A*, and
2. the intention that because of *S*'s utterance *H* not do *A*.

promise (promise, swear, vow; also contract, bet, swear that, guarantee that, guarantee x, surrender, invite) In uttering *e*, *S* promises *H* to *A* if *S* expresses:

1. the belief that his utterance obligates him to *A*,
2. the intention to *A*, and
3. the intention that *H* believe that *S*'s utterance obligates *S* to *A* and that *S* intends to *A*.

question (ask, inquire, interrogate, query, question, quiz) In uttering *e*, *S* questions *H* as to whether or not *C* if *S* expresses:

1. the desire that *H* tell *S* whether or not *P*, and
2. the intention that *H* tell *S* whether or not *C* because of *S*'s desire.

requestive (ask, beg, beseech, implore, insist, invite, petition, plead, pray, request, solicit, summon, supplicate, tell, urge) In uttering *e*, *S* requests *H* to *A* if *S* expresses:

1. the desire that *H* do *A*, and
2. the intention that *H* do *A* because (at least partly) of *S*'s desire.

requirement (bid, charge, command, demand, dictate, direct, enjoin, instruct, order, prescribe, require) In uttering *e*, *S* requires *H* to *A* if *S* expresses:

1. the belief that his utterance, in virtue of his authority over *H*, constitutes sufficient reason for *H* to *A*.
2. the intention that *H* do *A* because of *S*'s utterance.

retractive (abjure, correct, deny, disavow, disclaim, disown, recant, renounce, repudiate, retract, take back, withdraw) In uttering *e*, *S* retracts the claim that *C* if *S* expresses:

1. that he no longer believes that *C*, contrary to what he previously indicated he believed, and
2. the intention that *H* not believe that *C*.

retrodictive (recount, report) In uttering *e*, *S* retrodicts that *P* if *S* expresses:

1. the belief that it was the case that *P*, and
2. the intention that *H* believe that it was the case that *P*.

suggestive (conjecture, guess, hypothesize, speculate, suggest) In uttering *e*, *S* suggests that *P* if *S* expresses:

1. the belief that there is reason, but not sufficient reason, to believe that *P*, and
2. the intention that *H* believe that there is reason, but not sufficient reason, to believe that *P*.

suppositive (assume, hypothesize, postulate, stipulate, suppose, theorize) In uttering *e*, *S* supposes that *P* if *S* expresses:

1. the belief that it is worth considering the consequences of *P*, and
2. the intention that *H* believe that it is worth considering the consequences of *P*.

C Proposition compatibility

This appendix defines what it means for the *P* of an $F(P)$ structure to be *locutionary compatible* with the *F*. This topic is introduced by Bach & Harnish [4, p. 11] and refers to the concept that, for example, if a person predicts something, then that something has to be in the future; if a person retrodicts something, it has to have occurred in the past; etc. The last term below is a catch-all term which indicates that anything that does not violate the previous terms can be considered to have met this requirement.

```

propositionCompatible(ascibe, C, relatedTo(F, 0)).
propositionCompatible(describe, C, is(0, F)).
propositionCompatible(predict, C, time(T, P)) :-
    C = time(T, P),
    compareTime(T > now).
propositionCompatible(question, C, isTrueFalse(P)).
propositionCompatible(retrodict, C, time(T, P)) :-
    C = time(T, P),
    compareTime(T < now).
propositionCompatible(C, C).

```

D Definition of standard perlocutionary effects for all forces

This appendix defines the standard perlocutionary effects for all illocutionary forces implemented in this formal language. These are a formalization of the definitions given in Appendix B. The names of the forces listed here are the same as those given in that appendix except for **deny** whose name has been changed to **denial** in the formal definition.

D.1 Terms used in definitions

The following terms are used in the definitions.

- **standardEffects M** — defines the standard perlocutionary effects for message M.
- **flbcMsg()** — a standard FLBC message.
- **considerForKB X because Y** — consider adding X to the knowledge base because of Y.
- **S believes C** — it is the case that S believes that C.
- **S wants C** — S wants C.
- **do(S, C)** — S does C.
- **S permitTo C** — S is permitted to C.
- **S prohibitFrom C** — S is prohibited from C.
- **S should C** — X should do C.

- S obligated C because Y — S is obligated to C because of Y.
- S requireTo C — S is required to C.
- **Time predicates** — 1) `time(T, C)`: During some time period T, C is (was, will be) true. 2) `time(after(T), C)`: After time T it will be the case that C. 3) `time(before(T), C)`: Before time T it was the case that C. 4) `now`: refers to the current date and time.
- `determine(X, Y)` — the value of X is determined to be Y.
- `flbcMsg(H, S, F, P)` — an FLBC message of the form F(P) from H to S.
- Other (`relatedTo()`, `is()`)

D.2 Definitions

```
standardEffects flbcMsg(speaker=S, hearer=H, force=advise,
    content=C, context=X, id=ID) :-
    considerForKB S believes (H should C) because flbcMsgID=ID,
    considerForKB S wants (H believes (H should C)) because flbcMsgID=ID.
```

```
standardEffects flbcMsg(speaker=S, hearer=H, force=ascribe,
    content=relatedTo(F, O), context=X, id=ID) :-
    considerForKB S believes relatedTo(F, O) because flbcMsgID=ID,
    considerForKB S wants (H believes relatedTo(F, O)) because flbcMsgID=ID.
```

```
standardEffects flbcMsg(speaker=S, hearer=H, force=assent,
    content=C, context=X, id=ID) :-
    considerForKB S believes C because flbcMsgID=ID,
    considerForKB S believes (H believes C) because flbcMsgID=ID,
    considerForKB S wants (H believes C) because flbcMsgID=ID.
```

```
standardEffects flbcMsg(speaker=S, hearer=H, force=assert,
    content=C, context=X, id=ID) :-
    considerForKB S believes C because flbcMsgID=ID,
    considerForKB S wants (H believes C) because flbcMsgID=ID.
```

```
standardEffects flbcMsg(speaker=S, hearer=H, force=concede,
    content=C, context=X, id=ID) :-
    considerForKB S believes C because flbcMsgID=ID,
    considerForKB time(before(now), S believes not C) because flbcMsgID=ID,
    considerForKB S wants (H believes C) because flbcMsgID=ID.
```

```
standardEffects flbcMsg(speaker=S, hearer=H, force=confirm,
    content=C, context=X, id=ID) :-
```

```

considerForKB S believes C because flbcMsgID=ID,
considerForKB S wants (H believes C) because flbcMsgID=ID.

standardEffects flbcMsg(speaker=S, hearer=H, force=denial,
    content=C, context=X, id=ID) :-
    considerForKB S believes not C because flbcMsgID=ID,
    considerForKB S believes (H not believes not C) because flbcMsgID=ID,
    considerForKB S wants (H believes not C) because flbcMsgID=ID.

standardEffects flbcMsg(speaker=S, hearer=H, force=describe,
    content=is(0, F), context=X, id=ID) :-
    considerForKB S believes is(0, F) because flbcMsgID=ID,
    considerForKB S wants (H believes is(0, F)) because flbcMsgID=ID.

standardEffects flbcMsg(speaker=S, hearer=H, force=dispute,
    content=C, context=X, id=ID) :-
    considerForKB S not believes C because flbcMsgID=ID,
    considerForKB S believes (H believes C) because flbcMsgID=ID,
    considerForKB S wants (H not believes C) because flbcMsgID=ID.

standardEffects flbcMsg(speaker=S, hearer=H, force=dissent,
    content=C, context=X, id=ID) :-
    considerForKB S believes not C because flbcMsgID=ID,
    considerForKB S believes (H believes C) because flbcMsgID=ID,
    considerForKB S wants (H believes not C) because flbcMsgID=ID.

standardEffects flbcMsg(speaker=S, hearer=H, force=inform,
    content=C, context=X, id=ID) :-
    considerForKB S believes C because flbcMsgID=ID,
    considerForKB S believes (H not believes C) because flbcMsgID=ID,
    considerForKB S wants (H believes C) because flbcMsgID=ID.

standardEffects flbcMsg(speaker=S, hearer=H, force=offer,
    content=C, context=X, id=ID) :-
    considerForKB S believes
        (S obligated (C if do(H, flbcMsg(H, S, effective(accept), C)))
            because flbcMsgID=ID)
        because flbcMsgID=ID,
    considerForKB S wants (C if do(H, flbcMsg(H, S, effective(accept), C)))
        because flbcMsgID=ID,
    considerForKB S wants (H believes
        (S obligated
            (C if flbcMsg(H, S, effective(accept), C))
            because flbcMsgID=ID))
        because flbcMsgID=ID,
    considerForKB S wants (H believe
        (S wants (C if do(H,
            flbcMsg(H, S, effective(accept), C))))
        because flbcMsgID=ID.

```

```

standardEffects flbcMsg(speaker=S, hearer=H, force=permit,
    content=C, context=X, id=ID) :-
    considerForKB S believes (H permitTo C) because flbcMsgID=ID,
    considerForKB S wants (H believes (H permitTo C)) because flbcMsgID=ID.

standardEffects flbcMsg(speaker=S, hearer=H, force=predict,
    content=time(T, C), context=X, id=ID) :-
    considerForKB S believes time(T, C) because flbcMsgID=ID,
    considerForKB S wants (H believes time(T, C)) because flbcMsgID=ID.

standardEffects flbcMsg(speaker=S, hearer=H, force=prohibit,
    content=C, context=X, id=ID) :-
    considerForKB S wants (H believes (H prohibitFrom C))
        because flbcMsgID=ID,
    considerForKB S wants not do(H, C) because flbcMsgID=ID.

standardEffects flbcMsg(speaker=S, hearer=H, force=promise,
    content=C, context=X, id=ID) :-
    considerForKB S believes (S obligated do(S, C)
        because flbcMsgID=ID)
        because flbcMsgID=ID,
    considerForKB S wants do(S, C),
    considerForKB H believes (S obligated do(S, C)
        because flbcMsgID=ID)
        because flbcMsgID=ID.

standardEffects flbcMsg(speaker=S, hearer=H, force=question,
    content=isTrueFalse(C), context=X, id=ID) :-
    considerForKB S wants do(H, determine(isTrueFalse(C), TruthValue)) because flbcMsgID=ID,
    considerForKB S wants do(H, flbcMsg(H, S, inform, TruthValue)) because flbcMsgID=ID.

standardEffects flbcMsg(speaker=S, hearer=H, force=request,
    content=C, context=X, id=ID) :-
    considerForKB S wants do(H, C) because flbcMsgID=ID,
    considerForKB S wants (H wants do(H, C)) because flbcMsgID=ID.

standardEffects flbcMsg(speaker=S, hearer=H, force=require,
    content=C, context=X, id=ID) :-
    considerForKB S wants (H believes (H requireTo C)) because flbcMsgID=ID,
    considerForKB S wants do(H, C) because flbcMsgID=ID.

standardEffects flbcMsg(speaker=S, hearer=H, force=retract,
    content=C, context=X, id=ID) :-
    considerForKB S not believes C because flbcMsgID=ID,
    considerForKB S believes (H believes time(before(now), S believes C)) because flbcMsgID=ID,
    considerForKB S wants (H not believes C) because flbcMsgID=ID.

standardEffects flbcMsg(speaker=S, hearer=H, force=retrodict,

```

```
content=time(T, C), context=X, id=ID) :-  
considerForKB S believes time(T, C) because flbcMsgID=ID,  
considerForKB S wants (H believes time(T, C)) because flbcMsgID=ID.
```

Acknowledgments

File: foundation2.[tex,pdf]. This is a draft. Do not cite without permission of the author.

References

- [1] ALLEN, J. F., AND KAUTZ, H. A. A model of naive temporal reasoning. In *Formal Theories of the Commonsense World*, J. R. Hobbs and R. C. Moore, Eds. Ablex Publishing Corp., 1985, ch. 7, pp. 251–268.
- [2] APPLE COMPUTER, INC. Apple Event registry: Standard suites. <http://dev2.info.apple.com/FTPIndices/App-3.html>, downloaded on August 13, 1996.
- [3] AUSTIN, J. L. *How To Do Things With Words*, 2nd ed. Harvard University Press, 1975.
- [4] BACH, K., AND HARNISH, R. M. *Linguistic Communication and Speech Acts*. MIT Press, 1979.
- [5] BAKER, S. Forget the huddled masses: Send nerds. *Business Week* (July 21, 1997), 110+.
- [6] BALDASSARI, M., AND BRUNO, G. PROTOB: An object oriented methodology for developing discrete event dynamic systems. In *High Level Petri Nets*, K. Jensen and G. Rozenberg, Eds. Springer-Verlag, 1991, ch. 25, pp. 624–648.
- [7] BARBUCEANU, M., AND FOX, M. S. COOL: A language for describing coordination in multi-agent systems. In *Proceedings of First International Conference on Multiagent Systems* (San Francisco, CA, June 12–14, 1995), AAAI Press, pp. 17–24.
- [8] BARBUCEANU, M., AND FOX, M. S. Coordinating multiple agents in the supply chain. In *Proceedings of the 5th Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (Stanford, CA, June 19–21, 1996), IEEE Computer Society Press, pp. 134–141.

- [9] BARBUCEANU, M., AND FOX, M. S. The design of a coordination language for multi-agent systems. In *Intelligent Agents III: Agent theories, architectures and languages* (Budapest, Hungary, August 12–13, 1996), ECAI, Springer-Verlag, pp. 341–55.
- [10] BORENSTEIN, N. S. Computational mail as network infrastructure for computer-supported cooperative work. In *Proceedings of the Conference on Computer-Supported Cooperative Work* (1992), J. Turner and R. Kraut, Eds., ACM SIGCHI & SIGOIS, ACM Press, pp. 67–74.
- [11] BRUNO, G., AND ELIA, A. Extending the entity-relationship approach for dynamic modeling purposes. In *Entity-Relationship Approach*, S. Spaccapietra, Ed. Elsevier Science Publishers, 1987, pp. 169–181.
- [12] CASTELLS, M. *The Rise of the Network Society*, vol. I of *The Information Age: Economy, Society, and Culture*. Blackwell Publishers, 1996.
- [13] COHEN, P. R., AND PERRAULT, C. R. Elements of a plan-based theory of speech acts. In *Readings in Distributed Artificial Intelligence*, A. H. Bond and L. Gasser, Eds. Morgan Kaufman, 1988, pp. 169–186.
- [14] COMPAQ COMPUTER CORPORATION. Compaq EDI home page. <http://www.compaq.com/corporate/EDI/>, downloaded on June 11, 1999.
- [15] COVINGTON, M. A. Speech acts, electronic commerce, and KQML. To appear in *Decision Support Systems*, 1997.
- [16] DARPA KNOWLEDGE SHARING INITIATIVE EXTERNAL INTERFACES WORKING GROUP. Specification of the KQML Agent-Communication Language. <http://www.cs.umbc.edu/kqml/kqmlspec.ps>, June 15, 1993. Downloaded on July 14, 1997.
- [17] FININ, T., FRITZSON, R., MCKAY, D., AND MCENTIRE, R. KQML as an agent communication language. In *Proceedings of the Third International Conference on Information and Knowledge Management* (Gaithersburg, Maryland, November 29–December 2, 1994), NIST, ACM Press, pp. 456–463.
- [18] FLORES, F., GRAVES, M., HARTFIELD, B., AND WINOGRAD, T. Computer systems and the design of organizational interaction. *ACM Transactions on Office Information Systems* 6, 2 (April 1988), 153–172.

- [19] FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS. FIPA 97 specification part 2: Agent communication language. <http://www.fipa.org>, November 28, 1997. Geneva, Switzerland.
- [20] GORDON, M. D., AND MOORE, S. A. Depicting the use and purpose of documents to improve information retrieval. Forthcoming in *Information Systems Research*, 1998.
- [21] GRICE, P. Meaning. In *Studies in the Way of Words*, P. Grice, Ed. Harvard University Press, 1989, pp. 213–223. Originally published in *Philosophical Review*, 1957, 66:377–88.
- [22] HAREL, D. Statecharts: A visual formalism for complex systems. *Science of Computer Programming* 8 (1987), 231–274.
- [23] HEUSER, C. A., AND RICHTER, G. Constructs for modeling information systems with Petri nets. In *Advances in Petri Nets*, G. Rozenberg, Ed. Springer-Verlag, 1992, pp. 224–243.
- [24] HURST, G., KIMBROUGH, S. O., MOORE, S. A., AND OLIVER, J. R. The next generation: Exploiting advanced protocols for electronic commerce. Presentation at workshop *Frontiers in Electronic Commerce: Experimental Systems for Communication, Coordination, and Negotiation*, sponsored by SEI Center for Advanced Studies in Management (affiliated with Wharton), Philadelphia, PA, February 13, 1992.
- [25] KIMBROUGH, S. O. On representing schemes for promising electronically. *Decision Support Systems* (1990), 99–121.
- [26] KIMBROUGH, S. O., AND HUA, H. On defeasible reasoning with the method of sweeping presumptions. *Minds and Machines* 1, 4 (November 1991), 393–416.
- [27] KIMBROUGH, S. O., AND MOORE, S. A. Message management systems: Concepts, motivations, and strategic effects. *Journal of Management Information Systems* 9, 2 (Fall 1992), 29–52.
- [28] KIMBROUGH, S. O., AND MOORE, S. A. On obligation, time, and defeasibility in systems for electronic commerce. In *Proceedings of the Hawaii International Conference on System Sciences* (1993), J. Nunamiaker, Jr., Ed., vol. III, University of Hawaii, IEEE Computer Society Press, pp. 493–502.

- [29] KIMBROUGH, S. O., AND MOORE, S. A. On automated message processing in electronic commerce and work support systems: Speech act theory and expressive felicity. *ACM Transactions on Information Systems* 15, 4 (October 1997), 321–367.
- [30] KIMBROUGH, S. O., AND MOORE, S. A. On the spanning hypothesis for EDI semantics. In *Proceedings of the 32nd Hawaii International Conference on Systems Sciences* (Wailea, Maui, Hawaii, January 5-8, 1999), University of Hawaii, IEEE Computer Society Press.
- [31] LABROU, Y. *Semantics for an Agent Communication Language*. PhD thesis, University of Maryland, Computer Science and Electrical Engineering Department, August 1996. UMI #9700710.
- [32] LABROU, Y., AND FININ, T. A semantics approach for KQML — a general purpose communication language for software agents. In *Proceedings of the Third International Conference on Information and Knowledge Management* (Gaithersburg, MD, November 29–December 2, 1994), National Institute of Standards and Technology, ACM Press, pp. 447–55.
- [33] LABROU, Y., AND FININ, T. A proposal for a new KQML specification. Downloaded from <http://www.cs.umbc.edu/kqml/> in January 1998 (Technical Report CS-97-03), February 3, 1997.
- [34] LEE, R. M. Bureaucracies as deontic systems. *ACM Transactions on Office Information Systems* 6, 2 (April 1988), 87–108.
- [35] LEE, Y. K., AND PARK, S. J. OPNets: An object-oriented high-level Petri net model for real-time system modeling. *Journal of Systems Software* 20 (1993), 69–86.
- [36] LEVINSON, S. C. *Pragmatics*. Cambridge University Press, 1983.
- [37] LIGHT, R. *Presenting XML*, 1st ed. SAMS.net Publishing, 1997.
- [38] LITMAN, D. J., AND ALLEN, J. F. A plan recognition model for subdialogues in conversations. *Cognitive Science* 11 (1987), 163–200.
- [39] MAIOCCHI, R., AND PERNICI, B. Time reasoning in the office environment. In *Office Systems: Methods and Tools* (1987), G. Bracchi and D. Tschritzis, Eds., IFIP TC 8/WG 8.4, North-Holland, pp. 223–246.

- [40] MALONE, T. W., GRANT, K. R., TURBAK, F. A., BROBST, S. A., AND COHEN, M. D. Intelligent information-sharing systems. *Communications of the ACM* 30, 5 (May 1987), 390–402.
- [41] MCCARTHY, J. Elephant 2000: A programming language based on speech acts. Downloaded from <http://www-formal.stanford.edu/~jmc/elephant.ps> on January 20, 1998, 1993.
- [42] MOORE, R. C. A formal theory of knowledge and action. In *Formal Theories of the Commonsense World*, J. R. Hobbs and R. C. Moore, Eds. Ablex Publishing Corp., 1985, ch. 9, pp. 319–358.
- [43] MOORE, S. A. Categorizing automated messages. *Decision Support Systems* 22, 3 (March 1998), 213–241.
- [44] MOORE, S. A. KQML & FLBC: Contrasting agent communication languages. Forthcoming in *International Journal of Electronic Commerce*, 2000.
- [45] MOORE, S. A. On conversation policies and the need for exceptions. In *Issues in Agent Communication*, F. Dignum and M. Greaves, Eds., Lecture Notes in Artificial Intelligence. Springer-Verlag, 2000. Forthcoming.
- [46] MOORE, S. A., AND KIMBROUGH, S. O. Message management systems at work: Prototypes for business communication. *Journal of Organizational Computing* 5, 2 (1995), 83–100.
- [47] OBERWEIS, A., AND LAUSEN, G. Temporal aspects in office information systems. In *Office Systems: Methods and Tools* (1987), G. Bracchi and D. Tschritzis, Eds., IFIP TC 8/WG 8.4, North-Holland, pp. 247–266.
- [48] OBJECT MANAGEMENT GROUP. CORBA 2.2/IIOP Specification. Downloaded from <http://www.omg.org/library/c2indx.html> in January 2000, December 1999.
- [49] PETERSON, J. L. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.
- [50] PINCI, V., AND SHAPIRO, R. An integrated software development methodology based on hierarchical colored Petri nets. In *High Level Petri Nets*, K. Jensen and G. Rozenberg, Eds. Springer-Verlag, 1991, ch. 26, pp. 649–666.

- [51] POLANYI, L. A formal model of the structure of discourse. *Journal of Pragmatics* 12 (1988), 601–638.
- [52] RICHTER, G., AND VOSS, K. Towards a comprehensive office model integrating information and resources. In *Advances in Petri Nets*, G. Rozenberg, Ed. Springer-Verlag, 1985, pp. 401–417.
- [53] SHOHAM, Y. Agent-oriented programming. *Artificial Intelligence* 60 (1993), 51–92.
- [54] STRAWSON, P. On referring. In *Readings in the Philosophy of Language*, J. F. Rosenberg and C. Travis, Eds. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1971, pp. 320–44. Originally published in *Mind*, LIX, no. 235 (1950).
- [55] WINOGRAD, T., AND FLORES, C. F. *Understanding Computers and Cognition*. Ablex Publishing Corp., Norwood, NJ, 1986.
- [56] ZISMAN, M. *Use of Production Systems for Modeling Asynchronous Concurrent Processes*. PhD thesis, The Wharton School, University of Pennsylvania, 1977.