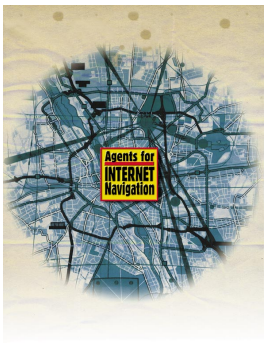
Agent Communication Languages: Rethinking the Principles



Agent communication languages have been used for years in proprietary multiagent systems. Yet agents from different vendors—or even different research projects—cannot communicate with each other. The author looks at the underlying reasons and proposes a conceptual shift from individual agent representations to social interaction.

Munindar P. Singh
North Carolina State University

Agents are important because they let software components interoperate within modern applications like electronic commerce and information retrieval. Most of these applications assume that components will be added dynamically and that they will be autonomous (serve different users or providers and fulfill different goals) and heterogeneous (be built in different ways). Agents can also be components themselves, which is characteristic of some promising modern systems.

Some entities are misrepresented as agents. The "agents" that marketing groups sometimes refer to, for example, are typically no more than glorified search engines or user interfaces. Such entities for the most part neither are aware of nor can communicate with other entities like them.[1] In the true sense of the word, an agent is a persistent computation that can perceive its environment and reason and act both alone and with other agents. The key concepts in this definition are *interoperability* and *autonomy*.

These concepts set agents apart from conventional objects, which always fulfill any methods invoked on them. Agents, in contrast, should be able to refuse an action. Thus, agents must be able to talk to each other to decide what information to retrieve or what physical action to take, such as shutting down an assembly line or avoiding a collision with another robot. The mechanism for this exchange is the *agent communication language*.

Theoretically, an ACL should let heterogeneous agents communicate. However, none currently do: Although ACLs are being used in proprietary multiagent applications, nonproprietary agents cannot interoperate. Many believe the fault lies in the lack of a formal semantics. Past efforts to standardize on the Knowledge Query Management Language, for example, failed because many dialects arose. The sidebar "Dialects and Idiolects" later explains in more detail how this can occur.

To provide agent interoperability, the Foundation for Intelligent Physical Agents is proposing a standard ACL based on France Télécom's Arcol. The hope is that Arcol's formal semantics will offer a rigorous basis for interoperability and prevent the proliferation of dialects. The sidebar "How Agent Communication Languages Have Evolved" describes how ACLs have attempted to realize these goals.

I believe this move toward a formal semantics is essential if ACLs are to unlock the full potential of agents. I am not convinced, however, that the existing work on ACLs, especially on the semantics, is heading in the right direction. It appears to be repeating the past mistake of emphasizing *mental agency*—the supposition that agents should be understood primarily in terms of mental concepts, such as beliefs and intentions. It is impossible to make such a semantics work for agents that must be autonomous and heterogeneous: This approach supposes, in essence, that agents can read each other's minds. This supposition has never held for people, and for the same reason, it will not hold for agents.

In this article, I show why an ACL's formal semantics should emphasize *social agency*. This approach recognizes that communication is inherently public, and thus depends on the agent's social context. I believe such an emphasis will help ease the fundamental tension between standardizing ACLs and allowing dialects. Both are desirable, but have thus far been mutually exclusive. A standard is needed to ensure that an ACL complies with a particular protocol; dialects

0018-9162/98/$10.00 © 1998 IEEE

How Agent Communication Languages Have Evolved

Figure A shows the progression of ACLs since the early days of agents, when there was little agent autonomy, and each project would invent its own ACL. The first significant interproject ACL was the Knowledge Query Management Language, proposed as part of the US Defense Advanced Research Projects Agency's Knowledge-Sharing Effort[1] in the late 1980s. Several KQML dialects are still being used.

KQML includes many primitives, all assertives or directives, which agents use to tell facts, ask queries, subscribe to services, or find other agents. A sample KQML message is `(tell :sender A :receiver B :content "raining")`. The semantics of KQML presupposes a virtual knowledge base for each agent. Telling a fact corresponds to reporting on that knowledge base; querying corresponds to the sending agent's attempt to extract something from the receiving agent's knowledge base.

In the early 1990s, France Télécom developed Arcol,[2] which includes a smaller set of primitives than KQML. Again, the primitives are all assertives or directives, but unlike KQML they can be composed. Arcol has a formal semantics, which presupposes that agents have beliefs and intentions, and can represent their uncertainty about various facts. Arcol gives performance conditions, which define when an agent may perform a specific communication. For example, in Arcol, agent Avi can tell agent Bob something only if Avi believes it also and can establish that Bob does not believe it. Arcol's performance conditions thus require the agents to reason about each other's beliefs and intentions and behave cooperatively and sincerely.

The most recent evolution of ACLs is the draft standard proposed by the Foundation for Intelligent Physical Agents (http://www.fipa.org/). The standard is heavily influenced by Arcol, adopting the Arcol model and semantics, although it softens a few of Arcol's performance conditions. The newer versions of the standard also discuss interaction protocols—a more promising line of thought. The FIPA standard also uses Lisp-like syntactic conventions similar to KQML's. For most purposes, however, the current FIPA standard can be treated the same as Arcol.

References

1. Y. Labrou and T. Finin, "Semantics and Conversations for an Agent Communication Language," in *Readings in Agents*, M. Huhns and M. Singh, eds., Morgan Kaufmann, San Mateo, Calif., 1998, pp. 235–242.
2. P. Breiter and M.D. Sadek, "A Rational Agent as a Kernel of a Cooperative Dialogue System: Implementing a Logical Theory of Interaction," in *Proc. ECAI-96 Workshop Agent Theories, Architectures, and Languages*, Springer-Verlag, Berlin, pp. 261–276.

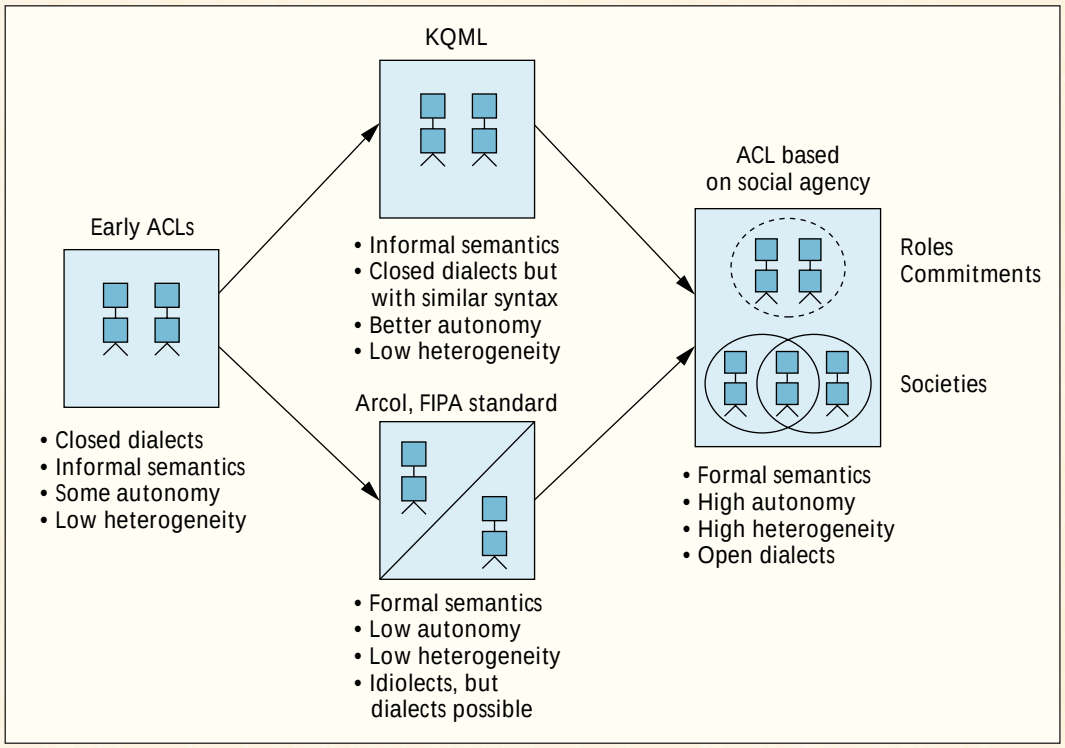


Figure A. ACL progression since the early days of agents.

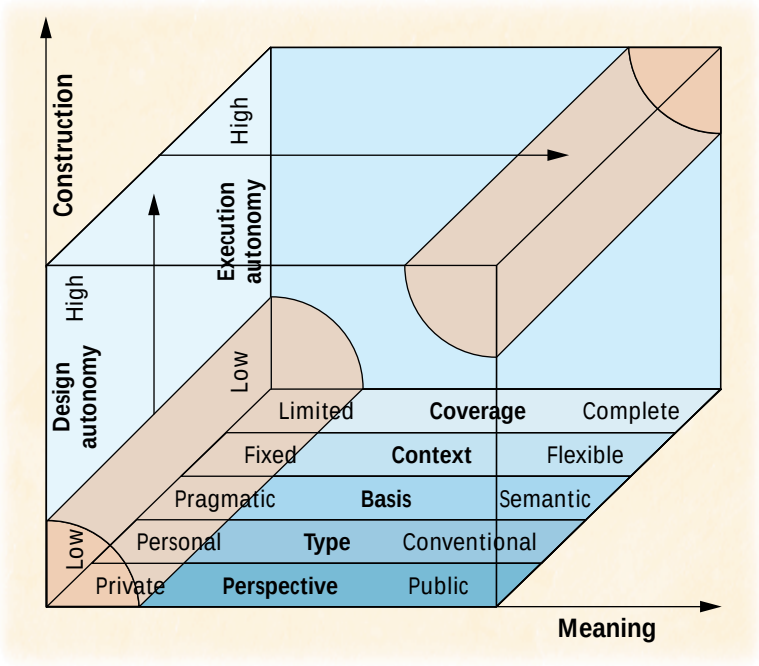


Figure 1. The design space of agent communication languages. The region in the lower left represents existing ACLs, which follow a mental agency model. The region in the upper right represents the desired goals, which dictate a social agency model: high design and execution autonomy, high coverage (includes all significant categories of communicative acts), flexible context, semantic basis for meaning, conventional meaning type, and a public perspective.

are needed to address the different scenarios that can arise with heterogeneous, autonomous agents.

In making the case for social agency, I look at the demands on an ACL and examine how KQML and Arcol are handling features along two critical dimensions: meaning and agent construction.

ELEMENTS OF MEANING

When agents function together, whether to cooperate or compete, they form a *multiagent system*. Multiagent systems provide higher level abstractions than traditional distributed programming. These abstractions are closer to user expectations and allow the designer more flexibility in determining behavior. For example, instead of hardwiring a specific behavior into the agents, multiagent system designers might have the agents negotiate with one another to determine the best course of action for that situation. Thus, ACLs must be flexible enough to accommodate abstractions such as negotiation. However, the same flexibility makes it harder to nail down their semantics.

For this reason, to arrive at the meaning of a communication you must examine many elements, including perspective, type of meaning, basis (semantics or pragmatics), context, and coverage (the number of communicative acts included).

Figure 1 shows the elements in this dimension. The region in the lower left characterizes existing ACLs, such as KQML and Arcol.

Perspective

Each communication has potentially three perspectives: the sender's, the receiver's, and the society's (that of other observers). The first two represent a *private* perspective. The third is a *public* perspective—the perspective of the multiagent system—available to all—as opposed to that of the individual agents.

Whose meaning should a language primarily reflect? As Figure 1 shows, both Arcol and KQML emphasize the private perspective. In fact, they are concerned only with the sender's perspective. This goes against the literature on human discourse (the very model that mental agency supposedly follows), which advocates treating the sender and receiver as equal partners.

For an ACL to be a true lingua franca, it must be *normative*—correctly designed agents must comply with the ACL so that agents from different design environments can understand each other. A normative ACL, in turn, must rely on some standard to ensure that different implementations preserve that ACL's meaning. To be effective, such a standard must provide some way to test for compliance. If an interaction breaks down, you should be able to determine which component failed (is not complying). If you cannot determine compliance, the standard is useless.

Furthermore, for compliance to be testable the ACL's semantics must have a public perspective. That is, it must emphasize social agency.

In fact, private perspectives are simply approximations of the public perspective. They merely have a role in determining how the agents decide what to communicate and how it is to be interpreted. An agent's designer may use the private perspectives, but only to set up the agent's beliefs and intentions so that its public behavior will comply with the standard.

Type of meaning

The formal study of language has three aspects. *Syntax* deals with how the symbols are structured, *semantics* with what they denote, and *pragmatics* with how they are interpreted and used. Meaning is a combination of semantics and pragmatics.

Pragmatics includes considerations external to the language proper, such as the mental states of the communicating agents and the environment in which they exist. Consequently, pragmatics can constrain how agents relate to one another and how they process the messages they send and receive. When the agents are

not fully cooperative or cannot determine implications as well as humans, they cannot meet the pragmatic requirements. If these requirements are an essential part of the ACL, no one can correctly apply it.

As Figure 1 shows, both Arcol and KQML emphasize pragmatics. In Arcol, an agent must make only *sincere contributions* (assertives that are believed true, requests that it intends should succeed) and may assume that other agents also make only sincere contributions. Consequently, you cannot use Arcol in settings where sincerity cannot be taken for granted—for example, in electronic commerce or, broadly, in negotiation of any kind.

Semantics versus pragmatics

A perspective can be combined with a type of meaning, either personal or conventional. In *personal* meaning, the meaning of communicative acts (described later) is based on the *intent* or interpretation of either the receiver or the sender. For example, the receiver may understand an act as a directive (purge this file) when it is syntactically an assertion (this is an old file) because the receiver is able to infer something from what the sender is saying.

Both Arcol and KQML emphasize a personal meaning, which can lead to problems. Even the recently proposed formalization of KQML[2] remains focused on personal meaning, although it considers the effect of a message on the receiver.

Consider Arcol's `inform` construct, which is supposed to merely give information. However, suppose an agent is to inform another agent that it is raining, but lacks either a belief in this statement or an intention to convey that belief to the receiving agent. Does an `inform` action take place? Traditional approaches offer no clear answer.

In *conventional* meaning, the meaning of communicative acts is based on usage conventions. The very idea of a lingua franca presupposes a well-defined conventional meaning. Indeed, language is nothing *but* a system of conventions, and they have proved to have considerable force. If you bid for an expensive item at Sotheby's, for example, you are liable for the price even if you didn't intend to pay.

By violating the idea of conventions, traditional approaches go against the wisdom of having different labels for communicative acts. KQML-based agents are notorious for replacing all their communicative acts with variants of the `tell` construct—KQML's version of Arcol's `inform`. Likewise, in Arcol, `requests` corresponds to `informs` of a certain kind. That is, if agent Avi is informed that agent Bob needs some information, Avi would supply that information as if Bob had requested it.

Thus, although traditional ACLs have different communicative acts, they are not capturing different conventions, but rather providing convenient abbreviations.

Context

In general, you cannot understand a communication without looking at its context—the agent's physical or simulated environment. Social context is central to the goals of an ACL. For agents, the social context need not be quite as subtle as it is for humans; it must determine only what agents expect of one another in their range of response, sincerity, and so on.

In general, you cannot understand a communication without looking at its context—the agent's physical or simulated environment.

As Figure 1 shows, both Arcol and KQML have a fixed context, partly because both languages have too many constraints and partly because they are inflexible. For example, by imposing the pragmatic requirement to be cooperative, Arcol requires the informing agent to believe the proposition being asserted is true; the informed agent to not already believe it; and the informer to intend that the informed agent come to believe it.

These requirements may not be acceptable in certain contexts. For example, suppose agent Avi wishes to repeat the conclusion of its negotiations with Bob with the phrase: "Okay, so the price is $5." Avi may communicate this only to formally conclude the negotiations even though it believes Bob already agrees. In Arcol, Avi would be unable to make this communication because it would violate a key prerequisite—that Avi believes Bob does not believe the price is $5.

Coverage of communicative acts

When heterogeneous, autonomous agents exchange information, the meaning of the exchange is characterized by *communicative acts*. For most computing scenarios, these acts fall into one of seven categories:

- *Assertives*, which inform: The door is shut.
- *Directives*, which request: Shut the door—or query: Can pelicans fly?
- *Commissives*, which promise something: I will shut the door.
- *Permissives*, which give permission for an act: You may shut the door.
- *Prohibitives*, which ban some act: You may not shut the door.
- *Declaratives*, which cause events in themselves: I name this door the Golden Gate.
- *Expressives*, which express emotions and evaluations: I wish this door were the Golden Gate.

Communicative acts can be put into a stylized form like "I hereby request . . . " or "I hereby declare . . . ". This grammatical form emphasizes that through language you not only make statements but also perform

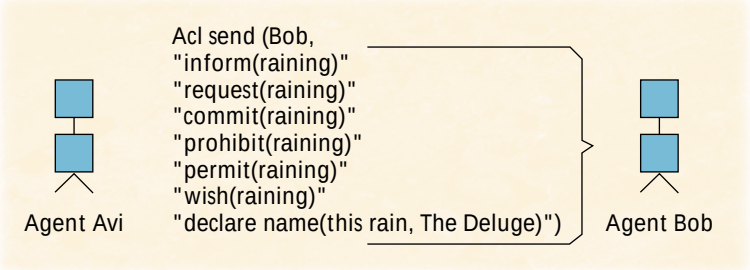


Figure 2. Why you cannot determine compliance under the mental agency model. You cannot determine whether agent Avi is compliant in sending the inform message, because there is no way to determine whether Avi believes it is raining. Similarly, the request is problematic, because there is no way to determine whether Avi believes Bob can make it rain. The same is true for the wish message. The other messages have a similar fate, although most wouldn't fit in traditional ACL syntax anyway.

actions. Acting by speaking becomes the essence of communication. For example, when a justice of the peace declares a couple man and wife, she is not only reporting their marital status, but also changing it. (For that reason, communicative acts are sometimes called *performatives*.)

As Figure l shows, Arcol and KQML have limited coverage; all primitives are either assertives or directives. In Arcol, you can simulate commissives using other acts. You can also reduce all acts to assertives, but using only the restricted meanings Arcol has for these categories. For example, a request in Arcol is the same as conveying to the receiver that the sender intends for it to perform the given action.

Although ACL designers should not try to anticipate all possible applications, they should be able to include acts from all seven categories because agents need them to enter into and manage more complex social relationships. Interacting with the underlying information system, for example, is important in many applications. This requires some way to initiate and maintain sessions, and authorize and commit to actions.[3] Commissives are essential for the agents to promise. Permissives and prohibitives let agents create or deny authority. Declaratives aid in appointing an agent as a representative or a group leader. And expressives let an agent convey evaluations and approvals. (This last category, although now rarely used, is likely to become more important when emotional agents become more common.)

AGENT CONSTRUCTION

Every ACL semantics must implicitly or explicitly embody some agent model. However, ACLs vary in what they emphasize (an individual agent's mental state or the social aspects of communication) and in how much design and execution autonomy they give an agent. Figure 1 shows how existing ACLs like KQML and Arcol handle the elements of agent construction.

Mental versus social agency

Mental agency emphasizes an agent's mental state, typically described as beliefs and intentions. Social agency regards agents as social creatures that interact with one another. As Figure 1 shows, both Arcol and KQML promote mental agency.

Mental states include

- *beliefs*, which characterize what an agent imagines its world state to be;
- *goals*, which describe what states the agent would prefer;
- *desires*, which describe the agent's preferences, sometimes with motivational aspects; and
- *intentions*, which characterize the goals or desires the agent has selected to work on.

Mental agency presupposes the *intentional stance*, which is the doctrine that you can describe any system using terms such as beliefs and intentions.[4] This is a compelling view because it says that modelers can create an agent using intentional terms. However, it does not solve the practical problem of how to determine the unique beliefs and intentions of an arbitrary agent just from its design and environment.

Consider the snippets of code for Agent Avi in Figure 2. How can you say whether or not Avi believes it is raining? Suppose you say that only agents with an explicit string `raining` in the data structure `beliefs` believe that it is raining. With that criterion, you eliminate a large subset of practical agents, because most agents do not carry a `beliefs` data structure. Moreover, if two agents did have a `beliefs` data structure and the structures were the same, the agents could act differently enough—because of differences in their programs or other data structures—that you couldn't say for sure whether they have the same beliefs.

On the other hand, without an explicit representation, anyone can claim anything about an agent's beliefs. For this reason, mental agency alone cannot provide the normative basis for an ACL semantics.

Each communicative act in Figure 2 poses a challenge for languages that promote mental agency. Traditional approaches also ignore whether Bob can really make it rain (when requested or permitted) or stop the rain (when prohibited); whether Avi can make it rain (when he promises); and whether Avi has the authority to permit or prohibit any of Bob's actions or to name weather conditions.

Ultimately, traditional ACL approaches conclude that if Avi's designer wants it to comply, it does. This is profoundly unsatisfactory, because it means that

compliance depends on neither the agent's behavior nor its design, but on how the design is documented. This position is conceptually and practically incoherent, because it means that any designer who cares to insert a comment saying "This program is correct" is freed from establishing its compliance.

A more promising approach is to consider communicative acts as part of an ongoing social interaction. Even if you can't determine whether agents have a specific mental state, you can be sure that communicating agents are able to interact socially. This is analogous to the distinction between an object's behavior (external) and state (internal). Interfaces in traditional software design are based on behavior, although state representations may be used to realize the desired behavior.

Practically and even philosophically, the compliance of an agent's communication depends on whether it is obeying the conventions in its society, for example, by keeping promises and being sincere.

Design autonomy

Design autonomy minimizes requirements on agent builders, thus promoting heterogeneity (the freedom to have agents of different design and construction). This, in turn, leads to a wider range of practical systems. For example, in a traditional setting, a Web browser can be implemented in any way as long as it follows the standard protocols.

Traditional approaches such as Arcol and KQML require agents to be implemented using representations of the mental concepts. As Figure 1 shows, this requirement reduces design autonomy. Agents may have to have beliefs and intentions, be able to plan and perform logical inferences, or be rational. These constraints also preclude many practical agent designs because you cannot uniquely determine an agent's mental state.

Execution autonomy

Execution autonomy corresponds to an agent's freedom to choose its own actions. An ACL can limit execution autonomy by requiring agents to be sincere, cooperative, benevolent, and so on. Execution autonomy is orthogonal to design autonomy because agents of a fixed design can have actions their designers cannot control; likewise, agents of diverse designs can have controllable actions. For example, two users with the same Web browser can still act differently, and those with different browsers can act the same if the browsers have similar functionality.

As Figure 1 shows, execution autonomy is low in Arcol; indeed, the language constrains agents to behave in ways many people could not emulate: Arcol agents must always speak the truth, believe each other, and help each other. This is appropriate for user interfaces—Arcol's original application domain—because the computational agent deals only with one other agent, the user. However, in other applications, this low autonomy means that Arcol can be applied only if the agent designers themselves subvert its semantics.

KQML, on the other hand, does not demand any specific form of sincerity or helpfulness and therefore better preserves execution autonomy. The historical reason for this difference is that KQML was designed for interoperation (although it failed), whereas Arcol was designed as a proprietary language for a specific system. Arcol designers reduced autonomy to suit that system, which simplified that system's design.

Our framework presupposes a richer infrastructure for agent management, which we term society management.

TOWARD SOCIAL PRINCIPLES

If, as Figure 1 shows, you assume that the ideal ACL would take a public perspective, emphasize conventional meaning, avoid pragmatics, consider context, and include all major communicative acts, you would be advocating a model that endorses *social agency*.

In an effort to move ACLs more closely toward that ideal, my colleagues and I at North Carolina State University are developing an approach based on societies of agents.

Protocols and societies

In our approach, agents play different roles within a society. The roles define the associated social commitments or obligations to other roles. When agents join a group, they join in one or more roles, thereby acquiring the commitments that go with those roles. The commitments of a role are restrictions on how agents playing that role must act and, in particular, communicate. In general, agents can operate on their commitments by manipulating or even canceling them.

These operations enable flexible behavior, but are themselves constrained by metacommitments to ensure that arbitrary behaviors do not result. Consequently, we specify protocols as sets of commitments rather than as finite state machines. Such protocol specifications can accommodate the kinds of exceptions that arise in multiagent systems.

Suppose that agent Avi is a seller and agent Bob is a buyer. Our protocol could include the following actions:

- Avi must respond to requests for price quotes (a form of cooperative behavior).
- Avi's price quotes issued to different agents within a specified period must be the same (sincerity).
- If Bob agrees to buy at a price, its check won't bounce (keeping promises).
- Avi will honor a price quote, provided Bob

responds within a specified period (keeping promises).

Designers can create specific protocols, and hence societies, for different applications such as electronic commerce, travel applications, industrial control, logistics, and student registration. As societies are designed, we envision that their specifications would be published.

Different vendors could supply agents to play different roles in these societies. Each vendor's agent would have to comply with the protocols in which it participates. Because protocol requirements would be expressed solely in terms of commitments, agents could be tested for compliance on the basis of their communications. This means the implementation need not be revealed, which is an important practical consideration (for example, to protect trade secrets). Also, because agents participate in a society, the society supplies the social context in which the communications occur. Thus, communicative acts can be more expressive and powerful because designers who agree on a standard society can assume a lot more about each other's agents.

Our framework presupposes a richer infrastructure for agent management, which we term *society management*. This infrastructure supports the definition of commitments, roles, and groups, as well as operations for agents to join a society in specific roles, to change roles, and to exit the society. Our framework also promotes execution autonomy. For example, Avi might only make assertions that it believes others don't already believe, whereas Bob may not restrict itself in such a manner. In general, the agents can act as they please provided they obey the restrictions of the societies they belong to and the protocols they follow.

Dialects and Idiolects

When agents from different vendors—or even different research projects—attempt to parse each other's messages, they cannot understand them correctly. This happens for two reasons. First, the receiving agent may not recognize the application-specific terms the sending agent is using to communicate. Second—and perhaps more important—even basic communication components are not uniformly understood. Both these problems stem from differing interpretations of key concepts, and the result is the evolution of multiple dialects within a language.

Idiolects—a variant of a language specific to one agent—result when the language emphasizes private perspective and personal meaning, as described in the main text. When only the private perspective is considered, an agent can produce or interpret messages as it unilaterally sees fit. Such an agent follows the philosophy of Lewis Carroll's Humpty Dumpty: Words mean exactly what I want them to. And communicating agents suffer the same problem as Alice, who failed to understand much of what Humpty Dumpty said.

Challenges

Our society-based approach avoids the problem of idiolects described in the sidebar "Dialects and Idiolects" because the essential semantic components act as normatives for agent behavior. Designers can create and popularize specialized societies—those that support more restrictive protocols for specific applications. When a protocol explicitly involves mental concepts (for example, by requiring a role to be sincere), it must also give some criteria to evaluate an agent's beliefs.

As such, our approach actually encourages dialects. The difference from the dialect problem described in the sidebar is that dialects in our approach have a social semantics and are not proprietary. Designers can define societies of their liking and implement agents to play appropriate roles in them. However, designers also know ahead of time the precise differences among dialects, and can expect their agents to communicate successfully with agents from other societies only to the extent that their dialects agree. Dialects of this variety enable the context sensitivity that is essential to building significant applications. Such dialects are good. The problem with traditional approaches is not the use of dialects per se, but that the dialects are arbitrary and cannot be adequately formalized in the chosen framework.

We envision the design and establishment of societies as essentially a community effort, much like Internet evolution. Protocols will spread much like the proliferation of network protocols, markup languages, and e-mail data formats: When enough vendors support a protocol, it will become a worthwhile target for other vendors.

The challenge thus becomes finding an approach that is normative at the society level and preserves some of the intuitions behind the high-level abstractions such as beliefs and intentions. Such an approach would provide a canonical form of protocols and a canonical definition for the different communicative acts. There are two obvious solutions. The first is to have a purely behavior-based approach, but this may limit the ability to describe complex agent states. The second is to have a purely mentalist approach, which as I have described, reduces agent design autonomy and is inherently noncanonical.

A third, less obvious, approach is to combine social commitments with a public perspective on the men-

tal states. This approach, which I originated and am currently investigating,[5] defines when an agent's communicative act would be wholeheartedly satisfied. For example, assertives are satisfied if the world matches what they describe. Directives are satisfied when the receiver acts to ensure their success—and has the required intentions and know-how. Commissives are satisfied when the sender acts to ensure their success. This approach is thus a hybrid: Although it takes some steps toward a coarse canonical set of objective definitions, it does not uniquely ascribe beliefs and intentions to agents. However, designers can use it to construct agents that would keep their public commitments.

Although all the fundamental issues in agent communication are far from resolved, my advice to people attempting to build multiagent systems is not to lose heart. Only through experience can some of these key questions be resolved. I have two recommendations. First, reflect on the issues this article raises as they affect a particular ACL or its implementation:

- What model of agency does the ACL require?
- How much does an ACL constrain an agent's design?
- Which perspective does the ACL embody?
- How can I determine what an agent believes or intends? You might need to make additional assumptions, essentially killing interoperation, to determine beliefs and intentions unambiguously. Alternatively, you might use beliefs and intentions only to design your own agents and not expect to know the details of other designs.

If, after answering these questions, you decide to use a specific ACL, understand that you have accepted its limitations as well. If the answers are unacceptable, you know to explore alternatives. A reasonable option is to reject the official semantics and use a commitment-based semantics instead.

My second recommendation is to start building systems. Always keep in mind that protocols are more important than individual communicative acts, and the best semantics is what you negotiate with other designers. For this reason, I believe that the strongest standards will develop in applications and markets that use agents heavily. As often happens in computing, the challenge will then be for the theoreticians to catch up. ❖

Acknowledgments

This is an extended and revised version of a position paper presented at the Fifth Meeting of the Foundation for Intelligent Physical Agents, April 1997. I thank Manny Aparicio, Michael Huhns, Yannis Labrou, Abe Mamdani, David Sadek, and Donald Steiner for discussions, the anonymous *Computer* reviewers for comments, and Nancy Talbert for careful editing.

This work is supported by the NCSU College of Engineering, the National Science Foundation under grants IIS-9529179 and IIS-9624425 (Career Award), and IBM Corp.

References

1. M. Huhns and M. Singh, "Agents and Multiagent Systems: Themes, Approaches, and Challenges," in *Readings in Agents*, M. Huhns and M. Singh, eds., Morgan Kaufmann, San Mateo, Calif., 1998, pp. 1-23.
2. Y. Labrou and T. Finin, "Semantics and Conversations for an Agent Communication Language," in *Readings in Agents*, M. Huhns and M. Singh, eds., Morgan Kaufmann, San Mateo, Calif., 1998, pp. 235–242.
3. M. Singh, J. Barnett, and M. Singh, "Enhancing Conversational Moves for Portable Dialogue Systems," in *Working Notes of the AAAI Fall Symp. Communicative Action in Humans and Machines*, American Assoc. of Artificial Intelligence, Menlo Park, Calif., 1997.
4. J. McCarthy, "Ascribing Mental Qualities to Machines," in *Formalizing Common Sense: Papers by John McCarthy*, V. Lifschitz, ed., Ablex Publishing, Norwood, N.J., 1990, pp. 93-118.
5. M. Singh, *Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications*, Springer-Verlag, Heidelberg, 1994.

Munindar P. Singh is an assistant professor of computer science at North Carolina State University, where his research interests are the theory and application of agents in information-rich environments. He is the editor-in-chief of IEEE Internet Computing *and a member of the editorial board of Kluwer's* Journal of Autonomous Agents and Multiagent Systems. *Singh is author of* Multiagent Systems *(Springer-Verlag, 1994) and coeditor of* Readings in Agents *(Morgan Kaufmann, 1998). Singh received a BTech in computer science and engineering from the Indian Institute of Technology, and an MSCS and a PhD in computer science from the University of Texas at Austin.*

Contact Singh at Dept. of Computer Science, North Carolina State University, Raleigh, NC 27695-7534; singh@ncsu.edu.