History, State of the Art and Challenges for Agent Communication Languages

Yannis Labrou and Tim Finin
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD 21250

ABSTRACT

Knowledge Query and Manipulation Language (KQML) is a language of typed messages, usually understood as *speech-acts*, encoded as ASCII strings (in a LISP-like syntax), that are transported over TCP/IP connections, and aimed at knowledge and information exchange between software systems that are viewed as *Virtual Knowledge Bases.* KQML, which first appeared almost 10 years ago, has come to define the concept of an ACL and in the process the ACL has become the centerpiece of a large category of agent systems. Inevitably an ACL has become a loosely-defined concept that encompasses a variety of issues which may or may not be ACL-relevant depending on one's point of view. The more "conservative" viewpoint advocates that the *semantics* of the message types is the one and only real issue. Agent development suggests though that *semantics* is the least important concern when one actually builds an agent system. The efforts of many researchers to develop multi-agent systems have brought to the foreground issues and considerations that are at least as important as the semantics for interoperable agent systems.

After introducing some of the basic concepts relating to Agent Communication Languages, we cover KQML and FIPA ACL, the two existing fully-specified ACLs. We give a brief introduction to their semantics and the issues relating to semantic descriptions of ACLs. We then shift our focus beyond the semantics and point to emerging threads of research in the ACL community. The issues that we deem relevant to the widest possible acceptance of ACLs include alternative syntactic encodings, services and infrastructure, integration with the WWW, and specification of conversation protocols.

1 Introduction

In the past ten years we have experienced a transformation of the term *agents* from a Artificial Intelligence term to a *buzzword*, often accompanied by promises of "amazing" feats that *agents* will be capable of. The term *agents* **—**by which we always mean *software agents*— *now* refers to a *paradigm* for software development that emphasizes autonomy both at design time and runtime, adaptivity, and cooperation; the agent paradigm seems appealing in a world of distributed, heterogeneous systems. This transformation has affected the tools and methodologies used for software agent development. A persistent theme throughout agents' conceptual evolution has been their ability to interact (communicate) with one another and thus be able to tackle collectively problems that no single agent can, individually.

For a large part of the agents' community, the role of endowing agents with the ability to communicate has been left to the *Agent Communication Language* (ACL). *Knowledge Query and Manipulation Language* (KQML), conceived in the early 90's gradually defined the concept of an ACL. KQML sprung out of the work of the *Knowledge Sharing Effort* (KSE), a consortium led by (mostly) AI researchers, which was aimed at achieving interoperability between knowledge bases. Early KQML can be summarized as a collection of *speech-act*-like message types, expressed as ASCII strings, with a LISP-like syntax, that are transported over TCP/IP connections, and aimed at knowledge and information exchange between software systems that are viewed as *Virtual Knowledge Bases*. KQML specifications and discussion about KQML manifested the AI-influenced thinking behind its design and the considerations that were deemed important, at that time.

In this article, we introduce some concepts useful in discussing agent communication languages and then compare and evaluate the two major ACLs. KQML, is our vehicle for introducing the fundamental notions of an ACL. The semantics of KQML have been the single most important issue in the debate over ACLs, and we include a brief overview of the relevant work. The second ACL we discuss is FIPA ACL. This is the language developed by the Foundation for Intelligent Physical Agents, the first organized effort focusing on developing standards in the broader area of agents. In our comparative evaluation of KQML and FIPA ACL, we look beyond the dominant issue of semantics and finally discuss thread of current (and desirable future) research in the broader ACL community.

2 Basic concepts of Agent Communication Languages

An ACL provides agents with a means of exchanging information and knowledge; Michael Genesereth has gone as far to equate agency with the ability of a system to exchange knowledge using an ACL (Genesereth and Ketchpel 1994). Of course, other means have been used to achieve the lofty goal of seamless exchange of information and knowledge between applications. From remote procedure call and remote method invocation (RPC and RMI) to CORBA and object request brokers, the goal has been the same. What distinguishes ACLs from such past efforts are the objects of discourse and their semantic complexity. ACLs stand a level above CORBA for two reasons:

- ACLs handle propositions, rules, and actions instead of simple objects with no semantics associated with them.

- An ACL message describes a desired state in a declarative language, rather than a procedure or method.

But ACLs by no means cover the entire spectrum of what applications might want to exchange. Agents can and should exchange more complex objects, such as shared plans and goals, or even shared experiences and long-term strategies.

At the technical level, when using an ACL, agents transport messages over the network using a lower-level protocol—for example, SMTP, TCP/IP, IIOP, or HTTP. The ACL itself defines the types of messages (and their meanings) that agents can exchange. Agents do not, however, just engage in single-message exchanges; they have *conversations*—task-oriented, shared sequences of messages that they follow, such as a negotiation or an auction. At the same time, a higher-level conceptualization of the agent's strategies and behaviors drives the agent's communicative (and non-communicative) behavior.

Traditionally, we understand the message types of ACLs as *speech acts*, which in turn we usually described in terms of beliefs, desires, intentions, and similar modalities. This kind of *intentional-level* description can be just a useful way to view a system, or it can have a concrete computational aspect. The second case describes a large range of BDI agents—*belief, desire, and intention* agents—that have some implicit or explicit representation of the corresponding modalities. This representation is built on top of a substrate that describes the conceptual model of the agent's knowledge, goals, and commitments, commonly known as a BDI theory.

3 Origins of Agent Communication Language Concepts

Understanding the evolution of the Agent Communication Language concept requires understanding the context that gave birth to KQML. KQML was first introduced as one of the results of the Knowledge Sharing Effort (KSE) (Neches, Fikes et al. 1991; Patil, Fikes et al. 1997), which has influenced ours and many other current efforts in inter-agent communication approaches.

The KSE was initiated as a research effort circa 1990 with encouragement and relatively modest funding from U.S. government agencies (DARPA especially). The KSE was highly active for roughly five years thereafter, and enjoyed the participation of dozens of researchers from both academia and industry; the researchers represented various branches of the AI community. Its goal was to develop techniques, methodologies and software tools for knowledge sharing and knowledge reuse between knowledge-based (software) systems, at design, implementation, or execution time. Agents, especially intelligent agents, are an important kind of such knowledge-based systems (other kinds include expert systems or databases, for example). The central concept of the KSE was that knowledge sharing requires communication, which in turn, requires a common language; the KSE focused on defining that common language

In the KSE model, agents (or, more generally, knowledge-based systems) are viewed as (virtual) knowledge bases that exchange propositions using a language that expresses various propositional attitudes. Propositional attitudes are three-part relationships between

- an agent,
- a content-bearing proposition (for example, it is raining), and

- a finite set of propositional attitudes an agent might have with respect to the proposition (for example, believing, asserting, fearing, wondering, hoping, and so on).

For example, $<a, fear, raining(t_{now})>$ is a propositional attitude.

The KSE model includes three layers of representation: (1) specifying propositional attitudes; (2) specifying propositions (i.e., "knowledge") - this is often called the (propositional) content layer; and (3) specifying the ontology (Gruber 1993) (i.e., vocabulary) of those propositions. The KSE accordingly includes a component (with associated language) for each of these: Knowledge Query and Manipulation Language (KQML) (Finin, Fritzson et al. 1992) for propositional attitudes, Knowledge Interchange Format[1] (KIF) (Genesereth and et.al. 1992) for propositions, and Ontolingua (Farquhar, Fikes et al. 1996) (which has supporting software tools) for ontologies.

Within the KSE approach, the three representational layers are viewed as mainly independent of another. In particular, the language for propositional content (i.e., the content language) can be chosen independently from the language for propositional attitudes. In other words, in the KSE approach, the role of an ACL, namely KQML's in the case of the KSE (or FIPA ACL's, which we discuss later) is only to capture propositional attitudes, regardless of how propositions are expressed, even though propositions are what agents will be "talking" about[2].

KQML was influenced by the Virtual knowledge Base concept which emphasized propositional content and focused on defining the propositional attitudes. KQML was not initially concerned with all of the "mechanics" of the interaction/communication nor prescribed much about how an agent is designed and how communication is incorporated in this design. These were intentional choices of the original KQML specification. The intent was to allow for various specific design choices on these issues. Even the chosen syntax was deemed as changeable.

4 KQML: concepts of ACL

Existing ACLs are KQML, its many dialects and variants, and FIPA ACL. KQML illustrates the basic concepts of all these.

KQML is a high-level, message-oriented communication language and protocol for information exchange independent of content syntax and applicable ontology. Thus, KQML is independent of the transport mechanism (TCP/IP, SMTP, IIOP, or another), independent of the content language (KIF, SQL, STEP, Prolog, or another), and independent of the ontology assumed by the content.

Conceptually, we can identify three layers in a KQML message: content, communication, and message:

- The content layer bears the actual content of the message in the program's own representation language. KQML can carry any representation language, including

[1] http://logic.stanford.edu/kif/ and http://www.cs.umbc.edu/kif/

[2] In a similar spirit, the approach of the technical committee that worked on FIPA ACL is that the content language should be viewed as orthogonal to the rest of the ACL message type.

languages expressed as ASCII strings and those expressed using binary notation. Every KQML implementation ignores the content portion of the message, except to determine where it ends.

- The communication layer encodes a set of features to the message that describe the lower-level communication parameters, such as the identity[3] of the sender and recipient, and a unique identifier associated with the communication.

- The message layer, which encodes a message that one application would like to transmit to another, is the core of KQML. This layer determines the kinds of interactions one can have with a KQML-speaking agent. The message layer's function is to identify the speech act or *performative* that the sender attaches to the content. This speech act indicates whether the message is an assertion, a query, a command, or any other of a set of known performatives. (KQML has adopted the term performative to mean any of its primitive message types). In addition, since the content is opaque to KQML, the message layer also includes optional features that describe the content language, the ontology it assumes, and some type of description of the content, such as a descriptor naming a topic within the ontology. These features make it possible for KQML implementations to analyze and properly deliver messages whose content is inaccessible.

4.1 Syntax and performatives

The syntax of KQML is based on the familiar s-expression used in Lisp—that is, a balanced parenthesis list. The initial element of the list is the performative; the remaining elements are the performative's arguments as keyword/value pairs. Because the language is relatively simple, the actual syntax is not significant and can be changed if necessary in the future. The syntax reveals the roots of the initial implementations, which were done in Common Lisp; it has turned out to be quite flexible.

A KQML message from agent ***joe*** representing a query about the price of a share of IBM stock might be encoded as shown in Figure 1a. In this message, the KQML performative is ***ask-one***, the content is ***(PRICE IBM ?price)***, the ontology assumed by the query is identified by the token ***NYSE-TICKS***, the receiver of the message is to be a agent identified as *stock-server*, and the query is written in a language called ***LPROLOG***. The value of the :content keyword is the content layer; the values of the ***:reply-with***, ***:sender***, and ***:receiver*** keywords form the communication layer; and the performative name with the ***:language*** and :***ontology*** keywords form the message layer. In due time, ***stock-server*** might send ***labrou*** the KQML message in Figure 1(b).

```
(ask-one
        :sender       labrou
        :content      (PRICE IBM ?price)
        :receiver     stock-server
        :reply-with   ibm-stock
        :language     LPROLOG
```

[3] The "identity" is some symbolic name, e.g., "labrou". One of the first identified problems was that these symbolic names provide no information about a program's actual network address and they can only be useful in the context of some existing name space, to which though there is no reference in the name (e.g., "labrou") itself.

```
        :ontology      NYSE-TICKS)
```
(a)

```
(tell
        :sender        stock-server
        :content       (PRICE IBM 114)
        :receiver      labrou
        :in-reply-to   ibm-stock
        :language      LPROLOG
        :ontology      NYSE-TICKS)
```
(b)

Figure 1. Examples of messages in KQML: (a) a query from agent joe about the price of IBM stock and (b) the stock-server's reply.

Although KQML has 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 original KQML specification considered the set to be extensible; a community of agents might choose to use additional performatives if they agree on their interpretation. However, an implementation that chooses to implement one of the reserved performatives must implement it in the agreed-upon way.

One of the design criteria for KQML was to produce a language that could support a wide variety of interesting agent architectures. Thus, KQML introduces a small number of performatives that agents use to describe capabilities; it also introduces a special class of agents called communication facilitators, which are (ordinary) KQML-speaking agents that are capable of processing the aforementioned set of performatives. A facilitator is an agent that performs various useful communication services, such as maintaining a registry of service names, forwarding messages to named services, routing messages based on content, matchmaking between information providers and clients, and providing mediation and translation services.

4.2 Semantics

During its first few years of use, KQML existed with only an informal and partial semantic description. Critics identified this as one of its shortcomings (Cohen and Levesque 1995). During the past few years, researchers have put forth several efforts to provide a formal semantics.

In other works (Labrou and Finin 1994; Labrou 1996; Labrou and Finin 1997; Labrou and Finin 1998), Labrou and Finin provide the semantics of KQML in terms of preconditions, postconditions, and completion conditions for each performative.

Assuming a sender A and a receiver B, preconditions indicate the necessary states for an agent to send a performative, ***Pre(A)***, and for the receiver to accept it and successfully process it, ***Pre(B)***. If the preconditions do not hold, the most likely response will be one of the performatives *error* or *sorry*.

Postconditions describe the states of the sender after the **successful** utterance of a performative, and of the receiver after the receipt and processing of a message but before a counter-utterance.

Postconditions ***Post(A)*** and ***Post(B)*** hold unless a sorry or an error is sent as a response to report the unsuccessful processing of the message.

A completion condition for the performative, ***Completion***, indicates the final state, after, for example, a conversation has taken place and the intention associated with the performative that started the conversation has been fulfilled.

Establishing the preconditions for a performative does not guarantee its successful execution and performance. The preconditions only indicate what can be assumed to have been the state of the interlocutors involved in an exchange, just before it occurred (assuming conforming interlocutor agents). Similarly, the postconditions describe the states of the interlocutors assuming the **successful** performance of the communication primitive. Preconditions, postconditions, and completion conditions describe states of agents in a language of mental attitudes (belief, knowledge, desire, and intention) and action descriptors (for sending and processing a message). No semantic models for the mental attitudes (BEL, WANT, KNOW, INT) are provided, but the language used to describe agents' states severely restricts the ways the mental attitudes can be combined to compose agents' states.

Figure 2 shows an example of semantics for **sender A** and **receiver B** in this framework. This semantics for *tell* suggests that an agent cannot offer unsolicited information to another agent. We can easily amend this by introducing another performative—let's call it *proactive-tell*—that has the same semantic description as *tell* but with ***Pre(A)*** being **BEL(A,X)**, and an empty ***Pre(B)***.

tell(A,B,X)

Pre(A): BEL(A,X) ∧ KNOW(A,WANT(B,KNOW(B,S)))
Pre(B): INT(B, KNOW (B,S))
where S may be any of BEL(B,X), or ¬(BEL(B,X)).
Post(A): KNOW (A,KNOW(B,BEL(A,X)))
Post(B): KNOW(B,BEL(A,X))
Completion: KNOW(B,BEL(A,X))

Figure 2 KQML semantics for *tell*.

Another semantic approach builds on earlier work on defining rational agency (Cohen and Levesque 1990; Cohen and Levesque 1990). The suggested approach views the language's reserved message types as attempts at communication. These attempts involve two or more rational agents that (temporarily) form "teams" to engage in *co-operative* communication. This approach strongly links the ACL semantics to the agent theory assumed for the agents involved in an ACL exchange.

5 Foundation for Intelligent Physical Agents

The Foundation for Intelligent Physical Agents is a nonprofit association whose purpose is to promote the success of emerging agent-based applications, services, and equipment. FIPA's goal is to make available specifications that maximize interoperability across agent-based systems. As this description suggests, FIPA is a standards organization in the area of software agents. The organization originally included the word physical in its name to cover agents of the robotic variety.

FIPA operates through the open international collaboration of member organizations, which are companies and universities active in the field. European and Far Eastern technology companies have been among the earliest and most active participants, including Alcatel, British Telecom, France Telecom, Deutsche Telecom, Hitatchi, NEC, NHK, NTT, Nortel, Siemens, and Telia.

FIPA's operations center around annual rounds of specification deliverables. The current specification is available at the FIPA home page, *http://www.fipa.org*. FIPA assigns tasks to technical committees, each of which has primary responsibility for producing, maintaining, and updating the specifications applicable to its tasks. The technical committee most important within the scope of this article is the one charged with producing a specification for an ACL. In addition, the agent management committee covers agent services such as facilitation, registration, and agent platforms; the agent/software interaction committee covers integration of agents with legacy software applications. Together, these three committees create the backbone of the FIPA specifications.

5.1 FIPA ACL

FIPA's agent communication language, like KQML, is based on speech act theory: messages are actions or communicative acts, as they are intended to perform some action by virtue of being sent. The FIPA ACL specification consists of a set of message types and the description of their pragmatics—that is, the effects on the mental attitudes of the sender and receiver agents. The specification describes every communicative act with both a narrative form and a formal semantics based on modal logic. It also provides the normative description of a set of high-level interaction protocols, including requesting an action, contract net, and several kinds of auctions.

FIPA ACL is superficially similar to KQML. Its syntax is identical to KQML's except for different names for some reserved primitives. Thus, it maintains the KQML approach of separating the outer language from the inner language. The outer language defines the intended meaning of the message; the inner, or content, language denotes the expression to which the interlocutors' beliefs, desires, and intentions, as described by the meaning of the communication primitive, apply. In FIPA ACL, the communication primitives are called communicative acts, or CAs for short. Despite the difference in naming, KQML performatives and FIPA ACL communicative acts are the same kind of entity[4].

5.2 Semantics.

SL is the formal language used to define FIPA ACL's semantics. SL is a quantified, multimodal logic with modal operators for beliefs (*B*), desires (*D*), uncertain beliefs (*U*), and intentions (persistent goals, *PG*). SL can represent propositions, objects, and actions. We can trace SL's origins to the work of Cohen and Levesque (Cohen and Levesque 1990), but its current form is primarily based on the work of Sadek (Sadek 1992). A detailed description of SL, including its own semantics, is outside the scope of this article and can be found in the FIPA ACL specification.

[4] To avoid confusion, we will use the terms performative, (communication) primitive, and communicative act interchangeably.

In FIPA ACL the semantics of each communicative act are specified as sets of SL formulae that describe the act's feasibility preconditions and its rational effect. For a given CA a, the feasibility preconditions FP(a) describe the necessary conditions for the sender of the CA. That is, for an agent to properly perform the communicative act a by sending a particular message, the feasibility preconditions must hold for the sender. The agent is not obliged to perform a if FP(a) holds, but it can if it chooses. A communicative act's rational effect represents the effect that an agent can expect to occur as a result of performing the action; it also typically includes specified conditions that should hold true of the recipient. The receiving agent is not required to ensure that the expected effect comes about and might indeed find it impossible. Thus, an agent can use its knowledge of the rational effect to plan what CA to perform, but it cannot assume that the rational effect will necessarily follow.

Conformance with the FIPA ACL means that when agent A sends CA x, the FP(x) for A must hold. The unguaranteed RE(x) is irrelevant to the conformance issue.

This introduction should be enough for a basic understanding of the example in Figure 3, which shows the specification of the communicative act *inform,* in which agent i informs agent j of content ϕ. The content of *inform* is a proposition, and its meaning is that the sender informs the receiver that a given proposition is true. According to this semantics, the sending agent

- holds that the proposition is true ($B_i(\phi)$);
- does not already believe that the receiver has any knowledge of the truth of the proposition ($\neg B_i(Bif_j(\phi) \vee Uif_j(\phi))$); and
- intends that the receiving agent should also come to believe that the proposition is true (rational effect $B_j(\phi)$);

$<i, inform(j, \phi)>$

FP: $B_i(\phi) \wedge \neg B_i(B_if_j(\phi) \vee U_if_j(\phi))$

RE: $B_j(\phi)$

Figure 3. FIPA ACL semantics for the communicative act *inform.* Agent *i* informs agent *j* of content ϕ.

6 Similarities and differences between KQML and FIPA ACL

KQML and FIPA ACL are almost identical with respect to their basic concepts and the principles they observe. The two languages differ primarily in the details of their semantic frameworks. In one sense, this difference is substantial: it would be impossible to come up with exact mappings or transformations between KQML performatives and their completely equivalent FIPA primitives, or vice versa. On the other hand, the ineluctable differences might be of little importance to many agents' programmers, if their agents are not true BDI agents.

Both languages assume a basic non-commitment to a reserved content language. However, in the FIPA ACL case, as we mentioned, an agent must have some limited understanding of SL to properly process a received message (as in the case of the *request* CA). The two languages have the same syntax. That is, a KQML message and a FIPA ACL message look syntactically identical—except, of course, in their different names for communication primitives. This is an important attribute of FIPA ACL. A large part of making an agent system communication-ready is to provide code that will parse incoming messages, compose messages for transport, and channel them through the network using a lower-level network protocol. This infrastructure will be the same regardless of the choice of ACL.

These encouraging thoughts do not apply to the semantics of the two languages. We can see that semantically the two languages differ at the level of what constitutes the semantic description: preconditions, postconditions, and completion conditions for KQML; feasibility preconditions and rational effect for FIPA ACL. They also differ at the level of the choice and definitions of the modalities they employ (the language used to describe agents' states). Although we can approximate the KQML primitives in FIPA's framework and vice versa, a complete and accurate translation is not, in general, possible. For example, to define a CA in FIPA ACL that approximates KQML's tell, we can replace ϕ in the definition of *inform* with $B_i\phi$.

Another difference between the two ACLs is in their treatment of the registration and facilitation primitives. These primitives cover a range of important pragmatic issues, such as registering, updating registration information, and finding other agents that can be of assistance in processing requests. In KQML, these tasks are associated with performatives that the language treats as first-class objects. FIPA ACL, intended to be a purer ACL, does not consider these tasks CAs in their own right. Instead, it treats them as requests for action and defines a range of reserved actions that cover the registration and life-cycle tasks. In this approach, the reserved actions do not have formally defined specifications or semantics and are defined in terms of natural-language descriptions.

The emergence of FIPA ACL might be a additional headache for implementers who must decide for themselves which one of the two ACLs to use. Any system that is to use KQML (or FIPA ACL, for that matter) must provide the following things:

1. a suite of APIs that facilitate the composition, sending, and receiving of ACL messages;

2. an infrastructure of services that assist agents with naming, registration, and basic facilitation services (finding other agents that can do things for your agent); and

3. code for every reserved message type (performative or communicative act) that takes the action(s) prescribed by the semantics for the particular application; this code depends on the application language, the domain, and the details of the agent system using the ACL.

Ideally, a programmer should only have to provide item 3. Items 1 and 2 should be reusable components that the programmer integrates into the application code. Actually, the programmer should not even have to integrate the listed under item 2; they ought to exist as a continuous running service available to any new agent. Currently though, no service exists through which one can register an agent by just sending a registration message. The disagreement over such services and agent naming conventions has resulted to a multitude of APIs. Every multi-agent system that uses an ACL has a homegrown implementation of these APIs—there are more than a handful of APIs written in Java, for Java agents—and its own infrastructure of basic services.

Providing the code that processes the primitives is more of an art than a science. The presented semantic approaches rely on multi-modal logics that are often non-computable or have no efficient implementation. The process of grounding the theory into code would result in a system that differs substantially—and probably unpredictably—from the theory on paper. To make matters worse, if the code does not implement at all the modalities assumed by the semantics, the programmer will most likely follow his or her intuitive understanding of the semantics of the communication primitives. These reasons have led to a growing interest in conversation protocols, a topic we discuss in Section 7.

7 Agent Communication Languages: Trends and Current Research

Many different groups have been involved in the design and implementation of multi-agent systems that support agent communication via an ACL. The experiences of these efforts lead as to identify three major trends:

- Java is rapidly becoming the language of choice. Implementing BDI agents with traditional AI languages is problematic enough, but we have little experience and fewer tools for doing so with object-oriented languages like Java. This raises again the problems of existing semantic approaches and conformance to them.

- Many of the new APIs support conversations, an intuitive way of structuring an agent's behavior and activities.

- Each implementation introduces its own variety of supporting agents and services for tasks such as naming, authentication, monitoring, and brokering.

Drawing from these trends we identify some areas of ongoing and future work in the broader area of Agent Communication Languages. The shared motivations of these threads of research are a desire to circumvent the difficulties with current semantic frameworks for conformance testing and practical design and implementations, and to achieve the widest possible acceptance of ACLs.

Syntax, Encoding and Pragmatic Considerations

The core semantics of an ACL is defined as the "deep" semantics (*i.e.*, semantics in the sense of declarative knowledge-representation) of its (communication) primitives. This semantics are expressed in some knowledge representation language: SL in the case of FIPA ACL. This semantics only takes into account the **speaker**, the **hearer** (in *speech act* terminology) and the content of the communicative act. The speaker, the hearer and the content correspond to the **:sender**, the **:receiver** and the **:content** of the *syntactic representation* of the ACL. The previous canonical syntactic form of the ACL message (for both KQML and FIPA ACL) is a Lisp-like ASCII sequence.

The (current) canonical ACL message syntax (both in FIPA ACL and KQML) further includes additional message parameters whose semantics go beyond that of the primitives. These parameters are unaccounted for in the deep semantics but are essential to the processing of an ACL message. One pragmatic aspect is parsing in and out of the ACL, *i.e.*, digesting and composing well-formed ACL syntax (which is Lisp-like) to extract or insert message parameters. A second pragmatic aspect is queuing (and de-queuing) ACL messages for delivery through TCP or some other network protocol. Further pragmatic issues being dealt with in the context of ACL efforts include the agent naming scheme, and the conventions for finding agents and initiating interaction; although, in our view, these issues are actually outside of the ACL's scope.

The various APIs for KQML and FIPA ACL provide nothing (as expected) regarding the actual processing of ACL messages (depending on the primitive), since the respecting the deep semantics of the primitives is the responsibility of the application that makes use of those API's. Such API's today mainly take care of the parsing and queuing tasks mentioned above. Performing these tasks is what using KQML (or FIPA ACL, for that matter) has come to mean. For all intents and purposes, compliance with the ACL's specification means compliance with all these pragmatic conventions. These conventions are not part of the standard (to the extent that the ACL semantics is standardized) and the subtle (or not so subtle) discrepancies amongst their implementations account in large part for the situation today in which there is often a lack of interoperability between systems using the same ACL.

We believe that an ACL has to have an abstract syntax. Fixed elements of the abstract syntax are the ones that have a direct semantic equivalent (sender, receiver, performative, content). Messages might have multiple encodings, Java objects, Lisp-like ASCII, etc. We advocate XML as a more interoperable and flexible encoding. XML can also assist with addressing the aforementioned pragmatics aspects and facilitate the integration of agents with the next wave of WWW technologies. We elaborate on these points below.

Services and Infrastructure

For the past few years many different groups have been involved with developing the services and infrastructure for ACL-communicating agents. To speak an ACL means to compose well-formed ACL messages, to be able to parse them, to send and receive them over the network and to have some scheme for naming agents. The differing approaches have to do in part with the lack of well defined and universally useful specifications for these services and in part due to the differing requirements of the implemented agent systems.

We do not expect a one-size-fits-all solution to these problems.

We believe it is imperative to look into WWW-like schemes for naming. Multiple encodings (or, XML encodings, as we hope) will help with the overhead of composing and parsing (because there are XML-processing tools).

Facilitators, name servers, and similar services are hard to standardize because solutions and approaches are often application/domain dependant. FIPA though, attempts to attack this problem in a more disciplined manner and if it manages to generate enough momentum for its chosen solutions, we might see the adoption of some reference implementations for such specialized services.

Conversations

Communicating agents do not randomly send ACL messages. The sent messages are causally related to one another and are transmitted in hopes of achieving some goal or in an attempt to perform some task, *e.g.*, bidding on some service or simply obtaining a piece of information. Although, the original KQML specification alluded to some intuitive ordering/sequencing of messages for basic tasks (such as querying) it was not until (Labrou and Finin 1994) that the notion of explicitly specifying such sequences became part of our thinking of an ACL. These sequences, known as protocols, or conversations (or some times as conversation protocols) describe expected sequences of messages for performing specific tasks.

Given the problematic nature of compliance with the ACL's semantic account, conversations might be more useful than the semantics in software-developing. Conversations shift the focus from the agent's internal workings to its observable behavior—the sequences of messages it sends to other agents. Agents can agree on a conversation protocol for a particular task—a negotiation or an auction, for example—and then engage in a scripted interaction. We do not suggest that this is a conformance test, but it might be useful for an agent designer to know that its interlocutors will engage in a scripted, pre-specified communicative behavior.

(Bradshaw, Dutfield et al. 1997; Labrou and Finin 1997) offer examples of using conversations to drive agent behavior. Also, FIPA is working on the specification of conversations to supplement the FIPA ACL description, although the formalism for conversations' specification is still being worked on.

Integration of ACLs with the World Wide Web

KQML and FIPA ACL have evolved at a considerable distance from the mainstream of Internet technologies and standards. There was no WWW when KQML first appeared (there were no agents, either, for that matter). XML-encoding (and maybe XML encoding of the content layer itself) (Grosof and Labrou 1999) is a first step towards a better integration of ACL-speaking agents with WWW infrastructure. To the extend that ACLs are driving a great part of the agent work it is reasonable to suggest that ACL work ought to integrate easily with the WWW and to be able to leverage WWW tools and infrastructure. This motivates us to give (and to advocate) an Extensible Markup Language (XML) encoding of ACL messages, as a first step towards this kind of integration.

We, and other groups have designed preliminary DTDs for KQML and FIPA ACL, and such DTDs have been made available to the FIPA community. Encoding ACL messages in XML offers some advantages that we believe are potentially quite significant.

- The XML-encoding is **easier to develop parsers for** than the Lisp-like encoding. The XML markup provides parsing information more directly. One can use the off-the-shelf tools for parsing XML - of which there are several competent, easy-to-use ones already available - instead of writing customized parsers to parse the ACL messages. A change or an enhancement of the ACL syntax does not have to result to a re-writing of the parser. As long as such changes are reflected in the ACL DTD, the XML parser will still be able to handle the XML-encoded ACL message. In short, a significant advantage is that the process of *developing or maintaining* a parser is much simplified.

- More generally, XML-ifying makes ACL more ``**WWW-friendly**'', which **facilitates Software Engineering** of agents. Agent development ought to take advantage and build on what the WWW has to offer as a software development environment. XML parsing technology is only one example. Using XML will facilitate the practical integration with a variety of Web technologies. For example, an issue that has been raised in the ACL community is that of addressing security issues, *e.g.* authentication of agents' identities and encryption of ACL messages, at the ACL layer. The WWW solution is to use certificates and SSL. Using the same approach for agent security considerations seems much simpler and more intuitive than further overloading ACL messages and the ACL infrastructure to accommodate such a task.

- As we mentioned earlier, the operational semantics of the pragmatic aspects of ACL can differ subtly between implementations or usages, and there is today a problem practically of interoperability. XML can help with these pragmatics, by riding on standard WWW-world technologies: to facilitate the engineering, and as a by-product to help standardize the operational semantics, thereby helping make interoperability really happen. Because XML incorporates links into the ACL message, this takes a significant step toward addressing the problem (or representational layer) of specifying and sharing the **ontologies** used in an ACL message's content. The values of the ACL parameters are not tokens anymore, but links that can point to objects and/or definitions.

- More generally, **links may be useful for a variety of other purposes**. For example, the `receiver` parameter might have a link to network location that provides information about the **agent's identity**: e.g., its owner, contact and administrative information, communication primitives that the agent understands, network protocols and ports at which it can receive messages, conversation protocols it understands, *etc.*. This type of information is necessary for a establishing an extended interaction with another agent and has to somehow be available to an agent's potential interlocutors. The same argument can be made about the other message parameters.

8 In Conclusion

Agent Communication Languages have followed a 10-year path of evolution. Our ideas about the specification of an ACL and the issues surrounding the development of ACL-speaking multi-agent systems owe much to KQML and its community of researchers and users but in recent years, FIPA has presented a more disciplined approach of dealing with these problems. FIPA does not have all the answers yet, and there are still challenges for the ACL community, some of which we outlined in this article. The long-lasting pre-occupation with the semantics should not distract the ACL community from the practical considerations of agent development. The major challenge we would like the ACL community to tackle, is the integration of ACL ideas with the domain of WWW technologies because we see the WWW as the natural space for innovative and interesting agent systems.

References

Bradshaw, J. M., S. Dutfield, et al. (1997). KAoS: Toward an Industrial-Strength Open Agent Architecture. Software Agents. J. M. Bradshaw, AAAI/MIT Press.

Cohen, P. R. and H. J. Levesque (1990). “Intention is Choice with Commitment.” Artificial Intelligence **42**(2--3): 213--261.

Cohen, P. R. and H. J. Levesque (1990). Persistence, Intention, and Commitment. Intentions in Communication. P. R. Cohen, J. Morgan and M. E. Pollack. Cambridge, MA, MIT Press**:** 33-69.

Cohen, P. R. and H. J. Levesque (1995). Communicative actions for artificial agents. Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS'95), AAAI Press.

Farquhar, A., R. Fikes, et al. (1996). The Ontolingua Server: A Tool for Collaborative Ontology Construction. Palo Alto, Stanford Knowledge Systems Laboratory.

Finin, T., R. Fritzson, et al. (1992). A Knowledge Query and Manipulation Language for Intelligent Agent Interoperability. Fourth National Symposium on Concurrent Engineering, CE & CALS Conference.

Genesereth, M. and R. F. et.Êal. (1992). Knowledge Interchange Format, Version 3.0 Reference Manual, Computer Science Department, Stanford University.

Genesereth, M. R. and S. P. Ketchpel (1994). “Software Agents.” CACM **37**(7): 48--53.

Grosof, B. and Y. Labrou (1999). An Approach to using XML and a Rule-based Content Language with an Agent Communication Language. IJCAI99, Workshop on Agent Communication Languages, Stockholm, Sweden.

Gruber, T. R. (1993). “A Translation Approach to Portable Ontology Specifications.” Knowledge Acquisition **2**: 199-220.

Labrou, Y. (1996). Ph.D. Dissertation, Semantics for an Agent Communication Language. Computer Science Department. Baltimore, MD, University of Maryland Baltimore County.

Labrou, Y. and T. Finin (1994). A semantics approach for KQML-a general purpose communication language for software agents. 3rd International Conference on Information and Knowledge Management, Gaithersburg, MD.

Labrou, Y. and T. Finin (1997). Semantics and Conversations for an Agent Communication Language. Readings in Agents. M. Huhns and M. Singh, Morgan Kaufmann Publishers**:** 235-242.

Labrou, Y. and T. Finin (1998). Semantics for an Agent Communication Language. Agent Theories, Architectures and Languages IV. M. Wooldridge, J. P. Muller and M. Tambe, Springer-Verlag.

Neches, R., R. Fikes, et al. (1991). “Enabling Technology for Knowledge Sharing.” AI Magazine **12**(3 (Fall)): 36--56.

Patil, R. S., R. E. Fikes, et al. (1997). The DARPA Knowledge Sharing Effort: Progress Report. Readings in Agents. M. Huhns and M. Singh, Morgan Kaufmann Publishers.

Sadek, M. D. (1992). A study in the logic of intention. Proceedings of the 3rd Conference on Principles of Knowledge Representation and Reasoning (KR'92). Cambridge, MA**:** 462-473.