Using Conceptual Graphs to Capture Semantics of Agent Communication

Lois W. Harper, Harry S. Delugach

Department of Computer Science
N300 Technology Hall
{lharper, delugach} @ cs.uah.edu
University of Alabama in Huntsville
Huntsville, AL 35899 USA

Abstract. Agent communication languages such as KQML and the FIPA ACL serve as metalanguages to define software agent message-passing protocols. These metalanguages are incompatible with each other, preventing intercommunication between agents employing different agent communication languages. The primary hindrance to agent intercommunication is the different underlying semantics of the message passing protocols. Conceptual graphs provide a mechanism to bridge this agent communication barrier by representing the semantics of message-passing protocols in the formal representation of conceptual graphs. Semantic content of the KQML **tell** performative is contrasted with that of the FIPA ACL **inform** performative and represented in conceptual graphs. The intent is that software agents conversant in CGIF may intelligently translate messages between agents employing different agent communication languages.

1 INTRODUCTION

A great deal of interest exists in developing enterprise agent-based software applications that will support distributed processes and distributed artificial intelligence. The development of distributed agent-based applications is being pursued both for Internet applications and for other networked environments, such as those found in manufacturing facilities. Three technical challenges that have been identified in supporting these types of agent-based applications are:

- reusability of agent components to support short development lifecycles.
- definition of frameworks that support heterogeneous computing platforms.
- standardization in agent communication languages and protocols.

The third issue, standardization in agent communication languages and protocols, has been studied in light of human communication and Speech Act Theories [1]. Shen, et al [2] state what is needed for intelligent agents to see practical application:

- agents need to be aware of the existence of other agents and of their access mechanisms.
- agents need to know and be capable of communicating their domain ontology.
- agents need to be aware of all available agent services.
- agents need to know how to monitor other agent tasks for completion success and performance status.
- agents need to know how to specify parameters to customize their service requests.

This list emphasizes that good communication is as critical for successful software agent applications as it is for any successful human endeavor. However, software agents are a new programming paradigm. There are no commonly accepted definitions of what comprises a software agent (mobile or intelligent), nor are software agent systems and agent communication mechanisms standardized. A significant number of different and incompatible agent-based systems will continue to exist until industries can agree upon and apply standards for agent interfaces, agent communication languages and agent computing platforms.

2 AGENT COMMUNICATIONS AND CONCEPTUAL GRAPHS

FIPA ACL and KQML are two Agent Communication Languages (ACLs) that model communication between intelligent software agents after spoken human communication. These models draw from speech act theories, particularly the approach presented by J. R. Searle [3]. Searle's theory considers natural language utterances such as questions, replies and declarations as actions, and also identifies three types of actions associated with an utterance:

- *Locution*, the physical utterance by the speaker
- *Illocution*, the intended meaning of the utterance by the speaker
- *Perlocution*, the action that results from the locution

The intended meaning of a communication act can be termed its *illocutionary force*. The term *performative* identifies the illocutionary force of an illocution and is identified by a verb such as "tell", "convince" or "ask" which is the core meaning of a speech act. The intended meaning of the message passed between two software agents corresponds to illocution in speech act theory. As a result, the messages between intelligent software agents are commonly referred to as *performatives*. The FIPA uses the terms *communicative act* instead of the term *performative*. Since *communicative act* and *performative* are interchangeable, we will use the term *performative* in this paper.

A performative is therefore a fundamental 'unit of communication' between software agents. A performative is a wrapper for some *content* that an agent wants to send or receive. The wrapper indicates who sent the message, what language the content of the message is written in, etc. Agent Communication Languages (ACLs) structure the type and sequence of performatives into communication *protocols* that

govern the exchange of messages between software agents. These ACL *protocols* are metalanguages that define both what are acceptable sequences of messages and also the expected outcome of acceptable sequences of messages. Thus a *performative* distinguishes between the type of message and its content, whereas a *protocol* is an expected sequence of messages.

The Foundation for Intelligent Physical Agents (FIPA) began to evaluate specifications for agent technologies in 1995. [12] Their purpose was to promote the success of emerging agent-based applications and services. By making specifications on agent-based technologies available to the global community, the FIPA intended to improve the ability of different agents implementations to operate as societies of agents capable of effective communication acts. [4] The FIPA determined that agent technologies that are grounded in artificial intelligence techniques were sufficiently mature for establishing specifications and FIPA specifications for agent technologies have been developed accordingly [5].

The Defense Advanced Research Project Agency (DARPA) Knowledge Sharing Effort (KSE) initiated research efforts towards knowledge sharing and reuse as early as 1990 [4]. DARPA produced several well-known software languages and utilities that facilitate intelligent agent communications, such as KQML, KIF, Ontolingua and OKBC. KQML was among the first of the ACLs to be developed and used [1], [2]. Researchers who have contributed to the DARPA KSE effort include Tim Finin and Yannis Labrou. A significant side-effect of the separate FIPA and DARPA KSE efforts towards defining and developing agent communication technologies is that the two most commonly known agent communication languages, KQML and the FIPA ACL (hereafter referred to as FIPA), cannot be directly translated from one form to the other [4].

As our modeling language we use conceptual graphs, which are a modeling and reasoning technique based around concepts and relations [6]. A special kind of relation, called an *actor*, is used to denote actions or functional relationships between concepts. With conceptual graphs (CGs), we can model semantics and then use our semantic models to reason about a domain. As an emerging knowledge interchange standard (through CGIF [7]) we can also exchange and re-use our models in other knowledge modeling efforts. The use of CG-based software agents that function as knowledge providers for Internet transactions is already being examined [8]. When conceptual graphs are used to represent the semantics of ACLs, a mechanism is provided that may allow software agents to communicate not only the content of a single message, but also the semantics of each performative and the intended outcome of a sequence of performatives. Since KQML and FIPA cannot be directly translated from one form to the other, their semantic framework must be implicitly represented in the knowledge bases of the software agents designed to be conversant in these languages. Capturing the semantics of these ACLs in CGIF form makes these semantic frameworks explicitly available to other software agents conversant in CGIF. The CGs presented in this paper were drawn using the CharGer Conceptual Graph Editor [13].

The main focus of this paper is to illustrate agent communication semantics in conceptual graphs by using a sample message sent from an agent called A to an agent called B. We then describe its semantics in some detail for each of the agent languages. Our purpose is two-fold: we want to show that conceptual graphs are

capable of representing agent communication semantics, and we want to use our representations to illustrate the semantic differences between the KQML and FIPA agent languages.

3 CONCEPTUAL GRAPH MODELS

This section presents conceptual graphs (CGs) that represent a **tell** KQML performative and an **inform** FIPA performative, and includes the following:

- a description of the **tell** KQML and **inform** FIPA performatives' syntax.
- a simple example proposition that forms the actual content to be communicated, represented in CG form.
- preconditions, postconditions and completion conditions for the **tell** KQML and **inform** FIPA performatives.

The goal of both the **tell** KQML performative and the **inform** FIPA performatives is to convey to some receiving agent that the sending agent believes a particular proposition (contained in the *content* field of a performative) is true. Referring to Table 1, it can be seen that the syntax of the FIPA **inform** performative is identical to KQML's **tell** performative (except for the two different keywords **tell** and **inform)**. The syntax of both FIPA and KQML is based on *s*-expressions in LISP; the first element is the performative name and the remaining elements are the keyword/value pairs that make up the performative's arguments. While the syntax of these FIPA and KQML performatives are identical, the semantic frameworks of the two ACLs are substantially different [4], which our example will illustrate. As a result, there is no exact transformation or mapping between the two ACLs, even in this simple example of two syntactically equivalent performatives that appear to support the same goal.

Table 1. KQML **tell** and FIPA **inform** Performatives

KQML tell Performative	FIPA ACL inform Performative
(tell : sender agent A : receiver agent B : content ("Agent A performs the task negotiate.") : language text)	(inform : sender agent A : receiver agent B : content ("Agent A performs the task negotiate.") : language text)

The FIPA and KQML ACLs do not impose any constraints on how a proposition is expressed in the content field. For example, the content may be statements in a programming language, a series of binary digits, or a text string as shown here. Consequently, the proposition in the content field may also be expressed in CGIF form.

A query in conceptual graphs is denoted by a graph where one or more concepts are unbound. We show bound and unbound CG representations of the sentence "*Agent A performs the task negotiate.*" in Figure 1. The unbound form of this sentence represents a query and is referred to as 'Y' in the KQML specification, but is not referred to by the FIPA ACL specification [9], [5]. The bound form of this sentence represents a proposition declared to be true by the agent sending the sentence. (The bound proposition is referred to as 'X' in both the KQML and FIPA ACL specifications.) In this paper, all references to the content field of this performative refer to the bound and unbound forms of the content sentence of Table 1, represented in CG form in Figure 1.

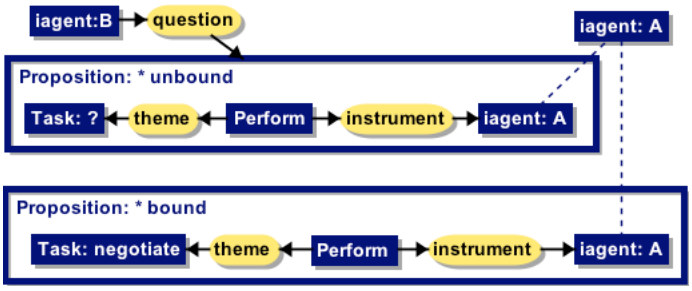


Fig. 1. CG Representation of a Proposition, unbound and bound Forms

We make the following note about the term *agent*. It is customary in CGs to use the relation name *agent* to link an action's concept to the animate-entity concept that performs the action [7]. Unfortunately, this may cause some confusion when the animate–entity itself is a software agent in the sense we are using in this paper. To prevent confusion, intelligent software agents will be denoted in the following CGs by the concept label *iagent*. Also, we will not use the relation *agent* in this paper.

3.1 Illocution of KQML tell and FIPA ACL inform Performatives

The CG representations of the KQML performative **tell** and the FIPA performative **inform** are shown in Figure 2. The left graph is the KQML performative **tell** and the right graph is the FIPA performative **inform**. These graphs are almst identical, as one would expect for performatives that are syntactically equivalent and appear to support the same goal. The goal of both performatives is to convey to the receiving agent (iagent B) that a sending agent (iagent A) believes the proposition in the performative's content field is true.

Looking at the two graphs in Figure 2, we see that they have identical structure and content. A key point of our paper is that their differences are found not in their structure, but in their semantics – their intent as well as the context in which they are used – as we will show in the rest of the paper.

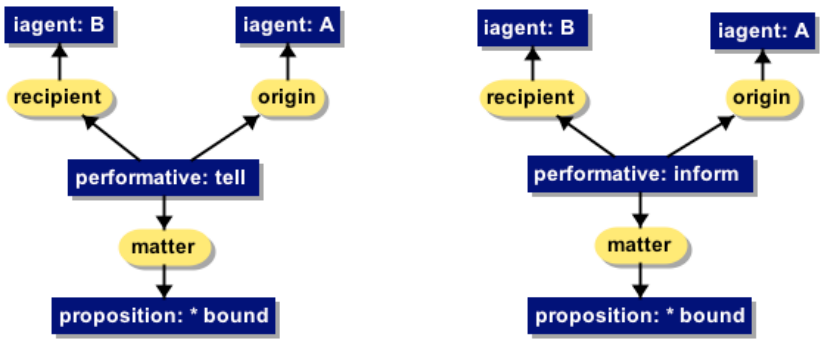


Fig. 2. CG Representation of the KQML **tell** (a) and FIPA **inform** (b) Illocution Act

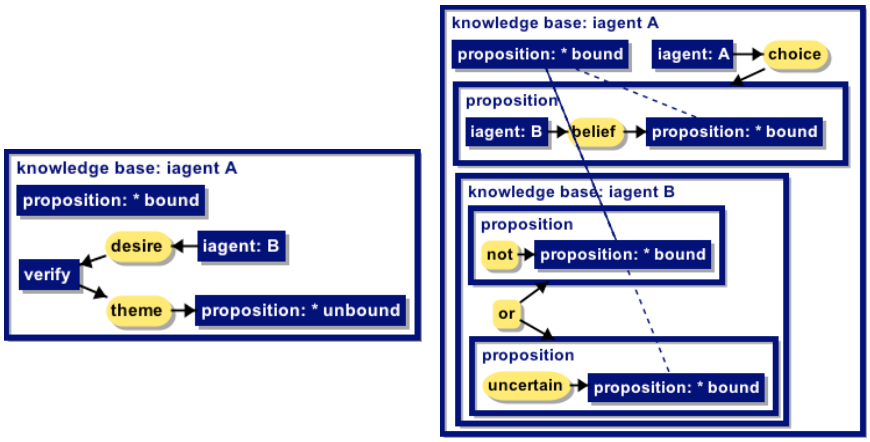
3.2 Preconditions on Agent A and Agent B

The semantics of KQML is expressed in terms of pre-conditions, post-conditions, and completion conditions for each performative [9]. If the pre- and post-conditions do not hold, a **sorry** or **error** performative is returned and the completion condition cannot occur. [9] A completion condition indicates that the intention, or core meaning, of the performative has been fulfilled.

FIPA expresses communication semantics in terms of a performative's feasibility conditions and its rational effect [5]. Feasibility preconditions are conditions necessary for the sender to perform the illocution act. The rational effect is the expected result(s) of the performative being fulfilled. Thus, in the FIPA specification for the **inform** performative, the feasibility preconditions are specified for the sender and rational effect postconditions are specified for the receiver. In our paper, we use the terms preconditions, postconditions and completion conditions for both the FIPA ACL **inform** performative and the KQML **tell** performative.

3.2.1 Preconditions on Agent A

The CGs shown in Figure 3 and summarized in Table 2 illustrate that both FIPA and KQML require that agent A holds the bound proposition to be true. However, KQML agent A (Figure 3(a)) also knows that KQML agent B has performed a query with the unbound proposition. The query may have been in the form of the KQML performatives **ask-if**, **ask-all** or **stream-all**. KQML agent A is not required to have prior knowledge of KQML agent B's knowledge base. FIPA agent A (Figure 3(b)) must intend that FIPA agent B comes to believe that the bound proposition is true, but also has two preconditions placed on its knowledge of FIPA agent B's knowledge base. FIPA Agent A must know that FIPA agent B has no knowledge of the bound proposition, or, that FIPA agent B is uncertain of the truth of the bound proposition.



(a) **(b)**

Fig. 3. CG Representation of Preconditions on KQML agent A **(a)** and FIPA agent A **(b)**

A practical consequence of these differences on the preconditions placed on agent A is that KQML agent A may send the **tell** performative whether or not KQML agent B has prior knowledge of the bound proposition, whereas FIPA agent A may not send the **inform** performative if FIPA agent A perceives that FIPA agent B has certain knowledge of the bound proposition. [5]

Table 2. Summary of Preconditions on agent A

KQML tell Preconditions (agent A)	FIPA inform Preconditions (agent A)
Agent A holds the bound proposition to be true	Agent A holds the bound proposition to be true
Agent A 'knows' that agent B desires to verify an unbound proposition	Agent B has no knowledge of the bound proposition or is uncertain of the bound proposition

3.2.2 Preconditions on Agent B

Although there are preconditions imposed on agent B in KQML [9] there are no preconditions specified for agent B in FIPA [5]. Figure 4 shows that KQML agent B has sent a query concerning the unbound proposition and desires a **tell** performative in return. The **ask-if** performative is represented in Figure 4, although the query sent may also have been one of the KQML **ask-all** or **stream-all** performatives.

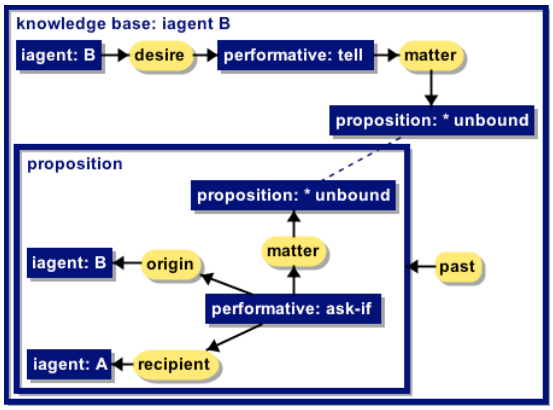


Fig. 4. CG Representation of KQML agent B's Preconditions

In contrast, Figure 5 shows that the FIPA **inform** performative enforces no preconditions on agent B. The FIPA Specification states, "… perhaps agent A has been asked," but it does not say that FIPA agent B has to have already placed a prior query with FIPA agent A. [5]

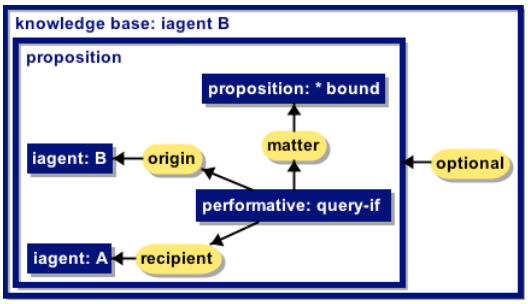


Fig. 5. CG Representation of FIPA agent B's Precondition

The FIPA specification [5] states that perhaps FIPA agent A has been asked about the proposition. As a result, this is an optional precondition that is not shown in Table 3.

Table 3. Summary of Preconditions on agent B

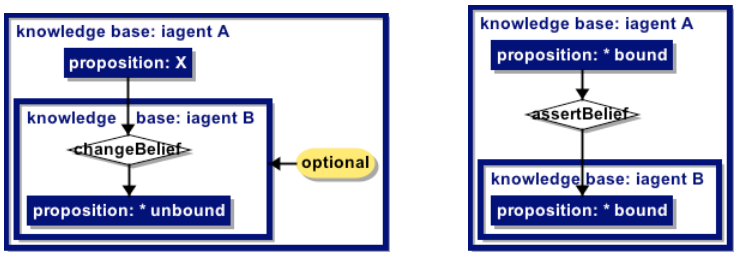
KQML tell Preconditions (agent B)	FIPA inform Preconditions (agent B)
Agent B has sent an 'ask-if', 'ask-all', or 'stream-all' performative to agent A in the past concerning the unbound proposition	(no mandatory preconditions)
Agent B desires to receive a tell performative concerning the unbound proposition	

3.3 Postconditions for the KQML tell and FIPA ACL inform Performatives

The CGs for the postconditions on agent A make use of the type of node referred to as an *actor*. Actors are used in CGs to represent processes that use their input concepts' referents to change the referents of their output concepts. [10] The actors in these CGs illustrate that the KQML **tell** performative and the FIPA ACL **inform** performative change agent A's view of agent B's knowledge base.

3.3.1 Postconditions on Agent A

A significant difference between the postconditions on agent A for the two ACLs is the effect the performative*s* have on agent A's representation of agent B's knowledge base. Referring to Figure 6 (summarized in Table 4), the viewpoint of KQML agent A (Figure 4(a)) is that the **tell** performative might have changed KQML agent B's unbound proposition to the bound proposition sent to KQML agent B by KQML agent A. The viewpoint of FIPA agent A (Figure 4(a)) is that the **inform** performative has asserted a bound proposition into FIPA agent B's knowledge base.



(a) **(b)**

Fig. 6. CGs For Postconditions on KQML agent A **(a)** and FIPA agent A **(b)**

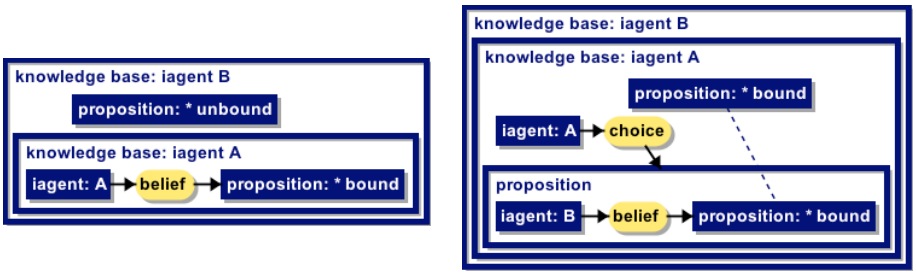
Bear in mind that these postconditions are representing agent A's viewpoint of the communication act, i.e., these postconditions reflect agent A's representation of what it knows about agent B's knowledge base.

Table 4. Summary of Postconditions on agent A

KQML tell Postconditions (agent A)	FIPA inform Postconditions (agent A)
Agent A holds the bound proposition to be true	Agent A holds the bound proposition to be true
Agent A's view of agent B's knowledge base shows that agent A has optionally changed the unbound proposition in agent B's knowledge base to the bound proposition.	Agent A's view of agent B's knowledge base shows that agent A has asserted the bound proposition into agent B's knowledge base.

3.3.2 Postconditions on Agent B

The postconditions for agent B are similar between the two ACLS. Figure 7(a) shows that KQML agent B is free to retain the unbound proposition (perhaps while polling a number of agents with the same query), but also knows that KQML agent A believes that the bound proposition is true. Figure 7(b) shows that FIPA agent B knows that FIPA agent A believes that the bound proposition is true, but is not required to accept the bound proposition sent by agent A. [5] states that " whether or not the receiver does, indeed, adopt the proposition will be a function of the receiver's trust in the sincerity and reliability of the sender." This is summarized in Table 5.



(a) **(b)**

Fig. 7. CGs For Postconditions on KQML agent B **(a)** and FIPA agent B **(b)**

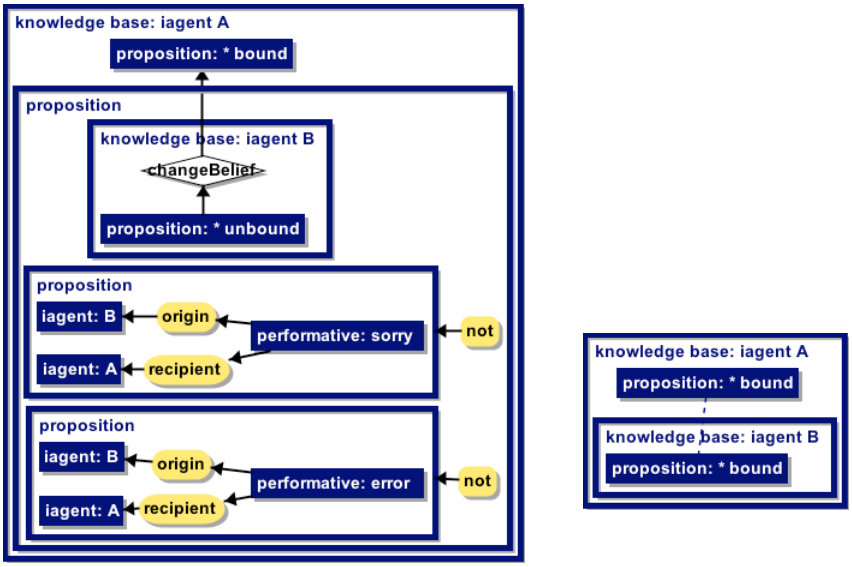
Table 5. Summary of Postconditions on agent B

KQML tell Postconditions (agent B)	FIPA inform Postconditions (agent B)
Agent B holds the unbound proposition	Agent B's view of agent A's knowledge base shows that agent A holds the bound proposition to be true.
Agent B's view of agent A's knowledge base shows that agent A believes the bound proposition is true.	Agent B's view of agent A's knowledge base shows that it is agent A's choice that agent B hold the bound proposition to be true.

3.4 Completion Conditions on Agent A for the KQML tell and FIPA ACL inform Performatives

Referring to Figure 8(a), the completion condition applies for KQML agent A as long as KQML agent B does not send a **sorry** or **error** performative. There is no statement as to 'how long' KQML agent A must wait for a **sorry** or **error** performative to be returned. [9] FIPA ACL does not state a completion condition, but does state that the

rational effect of the **inform** performative is that FIPA agent B (Figure 8(b)) believes the bound proposition sent by FIPA agent A. However, Section 3.3 noted that FIPA agent B is not required to accept the bound proposition sent by FIPA agent A. Our interpretation is that, whether or not FIPA agent B accepts the bound proposition, it is accepted in FIPA agent A's representation of FIPA agent B's knowledge base.



(a) **(b)**

Fig. 8. CGs For Completion Conditions on KQML agent B **(a)** and FIPA agent B **(b)**

4. DISCUSSION

Section 3 illustrated the different semantics underlying the KQML **tell** performative and the FIPA ACL **inform** performative. The difference in semantics is substantial although the performatives have the same syntax and goal (both intend to convey to a receiving agent that a sending agent believes a stated proposition is true).

The CGs of these performatives are shown in graphical form, but of course, they may also be represented in Conceptual Graph Interchange Form (CGIF) [7]. As the CGIF standard emerges, more systems will be able to share knowledge bases, so that intelligent software agents that are 'conversant' in CGIF may be written that are capable of bridging the communication gap between different ACLs.

Moore [11] analyzed the semantics of KQML and translated performatives into "more or less equivalent" forms in FLBC, the Formal Language for Business Communication. Moore's goal was not to facilitate communication between software agents conversant in these two ACLs. Rather, Moore states, "… a message for automated electronic communication should more closely reflect its underlying meaning" and proposes FLBC as "exemplar of this approach". The use of CGs to

capturing semantics of ACLs meets the need expressed by Moore to represent the underlying semantics of ACLs, but has the significant added advantage of representing both the explicit and implicit semantics of one or more ACLs in CGIF machine-readable form that offers interchange and reasoning capabilities beyond that of an ACL.

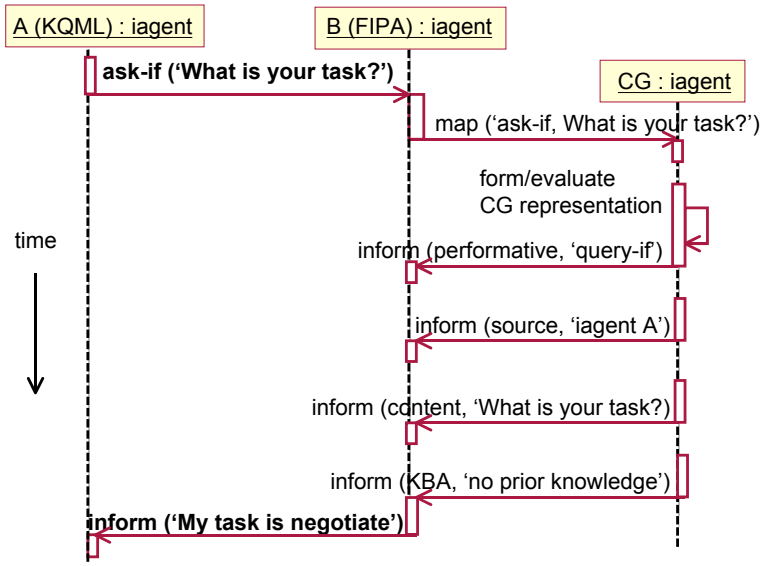


Fig. 9. Sequence Diagram Depicting Example Agent Performative Mapping Sequence

Figure 9 is a UML sequence diagram showing an example agent communication sequence. The purpose of a UML sequence diagram is to show the time-sequence of messages sent between objects. The messages sent between objects are depicted as horizontal lines. Time flows from the top of the diagram downward, although no time scale is implied. The objects at the top of Figure 9 are intelligent software agents.

The sequence diagram in Figure 9 shows an example agent communication scenario. In this example an agent conversant in FIPA receives an **ask-if** performative from an agent conversant in KQML. The content of the **ask-if** performative is the query, "What is your task?" The FIPA agent does not recognize the KQML **ask-if** performative. Since the FIPA agent is intelligent it may choose to ignore the message. However, consider the scenario where the services of a CG agent are available. The FIPA agent may refer the message to the CG agent for translation. This is shown in Figure 9 as a **map** performative. The CG agent evaluates the message, recognizes the performative as being a member of the KQML ACL and returns information that will assist the FIPA agent in responding to the KQML agent's message. In this example, the CG agent informs the agent conversant in FIPA that a similar FIPA ACL performative is query-if, and also informs the FIPA agent that the KQML agent has no prior knowledge of the FIPA agent's task. This is significant, since a practical consequence of the preconditions placed on agents is that a KQML agent may send the **tell** performative whether or not the receiving KQML agent has prior knowledge

of a proposition, whereas an FIPA agent may not send the **inform** performative if the sending FIPA agent perceives that the receiving FIPA agent has certain knowledge of a proposition. (Refer to Section 3.2.1) In this example scenario, the agent conversant in FIPA may then return an **inform** performative with the sentence "My task is negotiate."

Such behavior on the part of software agents requires that rules be developed for mapping the performatives in the different ACLs based on the semantic representation of the performatives. While this is not possible from analysis of the syntax of the performatives alone, the conceptual graph representation of the semantic content of a performative may form a basis from which this mapping process occurs.

The net effect of enlisting the mapping services of the CG-based agent is that messages may be sent between agents conversant in different, incompatible agent communication languages. An agent conversant in one ACL may send a performative to an agent conversant in a semantically different and incompatible ACL. The receiving agent may then return an appropriate response. In this scenario, indirect but effective translation of messages between agents conversant in different agent communication languages may occur.

Although this paper examined the semantics of two ACLs (FIPA and KQML), there are also many 'dialects' of KQML, each tailored to a specific application. [2] The same approach may be taken to communicate between agents conversant in different dialects of KQML and also may be useful in facilitating communication between legacy and current versions of the same dialect of an ACL.

Our future work is to examine the other performatives in both FIPA and KQML and represent their semantics in conceptual graph terms. This will give us an opportunity to improve our understanding of the performatives and their corresponding roles in FIPA and KQML ACLs. Our ultimate goal is to find ways that agents using different languages can automatically communicate, even if their underlying semantics differ.

5 CONCLUSIONS

In this paper, we have shown how conceptual graphs are capable of representing agent communication semantics, and we illustrated the semantic differences between two performatives used by two of the most commonly known agent communication languages, KQML and FIPA ACL. These performatives were selected because although they are syntactically equivalent and support the same communication goal, they are significantly different in their underlying semantics. The conceptual graphs shown illustrate specific semantic incompatibilities. We also illustrated a scenario of a possible communication sequence between agents conversant in two different ACLs.

Our aim is to overcome semantic incompatibilities by using CG-based translation techniques based on semantic representations in the machine-readable CGIF form. The conceptual graphs presented in this paper are meant as a proof-of-concept for the approach; we will pursue modeling of other performatives as well. Once we have established semantic models of other performatives used in the FIPA

and KQML ACLs, we will be in a position to develop CGIF-conversant agents that can facilitate the transfer of messages (and therefore message content) between agents designed for otherwise-incompatible ACLs.

REFERENCES

[1] M. N. Huhns and L. M. Stephens, "Multiagent Systems and Societies of Agents," *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, Cambridge: The MIT Press, 1999.

[2] W. Shen, N. Douglas, and J. P. Barthes, *Multi-Agent Systems for Concurrent Intelligent Manufacturing* New York, NY: Taylor & Francis Inc., 2001.

[3] J. R. Searle, *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press, Cambridge 1970.

[4] Y. Labrou, T. Finin, and Y. Peng, "Agent Communication Languages: The Current Landscape," *IEEE Intelligent Systems*, vol. 14, pp. 45-52, 1999.

[5] FIPA, "FIPA 97 Specification Part 2: Agent Communication Language," Foundation for Intelligent Physical Agents, October, 1998.

[6] J. F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*. Reading, Mass.: Addison-Wesley, 1984.

[7] J. F. Sowa, "Conceptual Graph Standard," http://users/bestweb.net/~sowa/cg/cgstandw.htm, 2000.

[8] G. W. Mineau, "A First Step Toward the Knowledge Web: Interoperability Issues Among Conceptual Graph Based Software Agents," in *Conceptual Structures: Integration and Interfaces*, vol. 2393, *Lecture Notes in Artificial Intelligence*, U. Priss, D. Corbett, and G. Angelova, Eds.: Springer-Verlag, 2002, pp. 250-260.

[9] Y. Labrou and T. Finin, "A Semantics Approach for KQML - A General Purpose Communication Language for Software Agents," presented at the Third International Conference on Information and Knowledge Management, 1994.

[10] H. S. Delugach, "Specifying Multiple-Viewed Software Requirements with Conceptual Graphs," *Journal on Systems and Software*, vol. 19, pp. 207-224, 1992.

[11] S. Moore, "KQML & FLBC: Contrasting Agent Communication Languages," *IEEE Proceedings of the 32nd Hawaii International Conference on System Sciences*, 1999.

[12] H. Suguri, "A Standardization Effort for Agent Technologies: The Foundation for Intelligent Physical Agents and Its Activities," *IEEE Proceedings of the 32nd Hawaii International Conference on System Sciences,* 1999.

[13] H. S. Delugach, " CharGer Conceptual Graph Editor," **http://www.cs.uah.edu/~delugach/CG**, 2003.