The Agent's Language

Communication in Multi-Agent Systems

Introduction

The most expressive and detailed way of communication is undoubly the usage of language. The power of words and speech did not only help the human being to develop but *created* the human being itself and its community. Higher-order problems which no individual being could solve on its own could be discussed and solved *together*. Following that idea it was time to construct a language for agents enabling them to tackle problems in a way like their human counterparts face them every day. The design of such an Agent Communication Language (ACL) depends on how to interpret what we call communication.

What are the characteristics of a language that serves communication? Obviously such a language must consist of tokens that are *shared* in a community where the participants communicate with each other. The sum of these shared tokens is also called the ontology, the body of background knowledge which is a particular conceptualization of a set of objects, concepts and other entities, as well as the relationship that hold among them. By defining this knowledge base which obviously is a presupposition for communication, the first ACL designing problem is faced. It cannot be expected that all agents in a multi-agent system really have the same ontology. In fact, this problem will always be present. It actually is present in human communities as well. But still communication is possible and communication is done all the time. So somehow it should be realizable to implement this ability in agents, too. Of course, ACLs are not only to copy human language using behaviour into multi-agent systems but also to give the agents a tool which should help finding (better) solutions for various problems. ACLs should make agents able to share knowledge and allow them to search independently for relevant information. The agents also ought to have the opportunity to work in teams or to find other agents who can assist them in order to fulfil a task. We see, the ontology an agent should entail, does not have to be totally complete concerning the world, but it should at least be able to manage these demands of an ACL.

How do ACLs work in principle?[1]

The recent approaches in agent communication languages are FIPA ACL and KQML (Knowledge Query and Manipulation Language), both using similar methods to provide the agent with a means to exchange information and knowledge: The agent transports the messages over a network by using a low-level protocol (e.g. SMTP, HTTP, TCP/IP), while the ACL itself defines the type of the message and the meaning that the agent may exchange. This allows a whole sequence of messages being bartered and conversations taking place between the agents. Simple agent communication protocols can be illustrated

[1] Cf. Finin and Fritzson, 1994. *KQML as an Agent Communication Language*, p. 2 – 4

with a client-server scenario: the client sends a query to the server, which will send back a reply to the client, if it is able to provide an answer or a set of answers. The server could also send back a handle which allows the client to ask for components of the reply, one at a time. Such client queries can be find e.g. in relational databases where the reply consists of a sequence of instantiations.

A more complex form of ACPs is the one with facilitators which perform communication services, like forwarding messages to certain named services. The facilitators are some kind of mediators which help the agents find an appropriate answer to their query. An agent might not always be able to ask another agent directly for information for the communication channel between them does not exist, or the agent is not allowed to do so. A facilitator can then be asked. A nice example for the employment of facilitators are programs managing auctions or stock exchange. There the messages can be routed according to their content, so the broker can get an individual reply or offer in a very short time.

KQML – Knowledge Query and Manipulation Language[2]

This ACL consists of a specific language and a set of such protocols. The characteristic of this language is its independency of the transport mechanism, i.e. the protocols used, and the content language, as well as the ontology which is assumed by the content. For KQML the content of a message is opaque. How is that possible?

KQML is built up of 3 layers. The *Content Layer* bears the actual content of a message, in the *Message Layer* the message is encoded that one application or agent would like to transmit to another, and the *Communication Layer* defines the action between them. The second layer makes up the core of KQML because it determines the kinds of interactions one can have with a KQML-speaking agent. Its primary function is to identify the network protocol to be used in order to deliver a message and to supply a performative which the sender agent attaches to the content. Of course, there can also be optional features in addition which e.g. describe the content language and the assumed ontology. Because of this layered structure it is possible for KQML implementations to analyse, route and properly deliver messages although their content is actually inaccessible.

This gets clearer when one looks at the syntax for KQML.

KQML Syntax[2]

The Syntax for KQML is mostly based on the s-expression which is known from Lisp. The specific structure begins with an initial element, the performative, and is continued by other elements like the performative's arguments in form of key-words or value pairs. The performatives are taken from a predefined set which is extendable.

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

Fig. 1) Agent Joe asks for the price of a share of IBM stock

```
(tell
      :sender stock-server
      :content (PRICE IBM 14)
      :receiver joe
      :in-reply-to ibm-stock
      :language LPROLOG
      :ontology NYSE-TICKS)
```

Fig. 2) The stock server sends a reply to Joe

In the figures above an example for the appearance of the syntax for KQML is shown. The initial element in Joe's query is "ask-one" whereas the rest defines the arguments given by the performative. The constraints and definitions can be seen in line two to seven: Joe is the sender, the content is a question concerning the price of an IBM stock, the receiver is the stock server, the message is composed in Prolog and the ontology is based on the New York Stock Exchange Ticks. The reply will be send back in respect to these lines as shown in Fig. 2.

KQML Semantics[2]

The fundamental semantic structure of communication between agents can be seen in terms of so called communicative acts, like e.g. informing, requesting, commanding, asserting, denying and many more. An ACL like KQML uses such acts to define the type of message one agent sends to another. Similar to human communication such acts demand conditions. That means, a certain performative is built up in a certain context. The agent who is going to perform an act of telling has to fulfil such conditions which are subdivided in *Pre-, Post- and Completion Conditions.* The precondition scenario for telling then is: *A knows X, B wants to know X.* So the precondition indicates the necessary state for agent A in order to send a performative and for the receiver B to accept it. If the preconditions

```
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 not(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))
```

Fig. 3) An example for the semantics of KQML;

[2] Cf. Labrou, Finin and Peng, 1999. The current landscape of Agent Communication Languages, p. 3 – 5

do not hold, the most likely response is "sorry" or "error".

Postconditions are taken to describe the state of the agents assuming the successful performance of the communicative utterance. They hold unless an error message is sent as a response. The telling example would have a postcondition scenario like this: *A told X to B, B knows X now.*

The completion conditions then indicate the final state, after the conversation has taken place and after the intention associated with the performative (that started the conversation) has been fulfilled.

In fig. 3) one can see how the semantics of KQML is realized with the pre- post- and completion conditions. If A wants to tell B something (X) then the precondition for A has to be that A believes X and that A knows that B wants to know X (here it can be seen how the substructure for *knowing* looks like: either it is that B wants to believe X or that B doesn't want to believe X. As an example one could take the information of "the door is open". So knowing X would result into a believe that X is true or that X is not true.) The precondition of B should be the intention of B to get the information concerning X. The postconditions are simply that the intention of B are fulfilled and that A is informed about the new knowledge of B. One can see that the completion condition is the same as the postcondition for B, namely that B now knows what A believes about X.

Problems

All these conditions describe the states of the communicating agents in a language of mental attitudes (belief, knowledge, intention, etc.). As mentioned before, the performatives are predefined and the set can be extended. But here problems similar to the one concerning the ontology are encountered. All agents have to own the same set of performatives in order to fulfil the conditions. In some way, ambiguities has to be cleared up. Some performatives might be defined vague for they are not even defined well in human language use.

Another point is that although the semantics of KQML and other ACLs are provided in terms of such mental attitudes, the agent itself does not need to reason using them, but it does only have to behave according to the principles they contain. This throws the language using agent miles away from the human using a language.

Fact is, that language *cannot* be described primitively in terms of performatives like it is done in ACLs nowadays. The speech act theory does not cover the whole spectrum of the humans' linguistic behaviour and the semantics of language. One of the very important factors left out, is *emotions*. If one tries to build up an ACL on the basis of the speech act theory, then there also have to be considered performatives which entail an emotional component or which *are* emotional throughout in a way.

Of course, in multi-agent systems the language does not play the role it does in human communication, nor does it have to. The dealing with information is the main point. This handling with knowledge should happen in an efficient and fast way, making it possible for agents to manage their tasks in a successful way. But that is not the main aim in human communication. So, when theories of human language usage are implemented in multi-agent systems, these two different aims collide in form of certain problems.

In contrast to the large critique of KQML, it should also be pointed out that there has been a strong improvement towards the older ACLs like COBRA or RMI. The semantics of KQML, although it lacks many features, is much more complex then the ones of the early approaches.

But agent communication languages should not only serve their own purpose, but develop towards the human linguistic performance. That development should make agents able to interact with human users in a much better way they already do.

It is interesting to see how some phenomena of human communication also occur in communication between agents in a multi-agent system: 'misunderstanding', false distribution of information, lacking knowledge, "sorry" as an empty reply to a demanding question. All these bugs in communication bring us back to the questions what communication between humans really is and what language is. And more: it gives the opportunity to analyze these phenomena and to find keys helping answer these fundamental questions.

In designing and improving the ACLs it is important to maintain an exchange of these two domains: implementing language theories into multi-agent systems and observing the development of the ACL. In order to do that, it still is crucial to become clear, what human language and communication really is and how we can define it.

Philosophical questions

KQML is bound so much to the predefined performatives which seem to form an obstacle for the deeper relevance that language should play in multi-agent systems. This doesn't reveal the lack of performatives or the wrong definition of them but a too much simplified assumption about language. How are we to take these concepts of belief, intention, and others in order to design a structure for an ACL? Does it seem sensible to use these mental attitudes in order to describe the states of an agent?

In many papers about KQML and ACLs in general the problems about the content of an utterance, let's say the information, and the language itself are not mentioned together but as separate parts. There is the content on the one side and the performative on the other. If we just take a look at the conditions in the semantics of KQML one can see that there is something in the brackets and something outside that puts the first argument in relation to the second (see fig. 3). Of course, this form is used in many programming languages and they describe propositions in an efficient and clear way. But how much does this appearance copy the structure of human language? And last but not least, the most important question: Where does *understanding language* find its place in all these considerations?

References

[1] T. Finin, and R. Fritzson, D. McKay and R. McEntire, 1994. *KQML as an Agent Communication Language*. In *The Proceedings of the Third International Conference on Information and Knowledge Management*, ACM Press, 1994.

[2] Y. Labrou, T. Finin and Y. Peng, 1999. *The current landscape of Agent Communication Languages.*

[3] Ian D. Craig, 1995. *A Perspective on Multi Agent Systems*, Chapter 3, p. 8 – 22.

[4] P. R. Cohen and H. J. Levesque, 1995. *Communicative Actions for Artificial Agents.*