Semantics for an Agent Communication Language

Yannis Labrou, and Tim Finin

Computer Science and Electrical Engineering Department
University of Maryland Baltimore County
Baltimore MD 21250 USA

{jklabrou,finin}@cs.umbc.edu

Abstract. We address the issue of semantics for an *agent communication language*. In particular, the semantics of Knowledge Query Manipulation Language (KQML) is investigated. KQML is a language and protocol to support communication between software agents. We present a semantic description for KQML that associates states of the agent with the use of the language's primitives (performatives). We have used this approach to describe the semantics for the whole set of *reserved* KQML performatives. Our research offers a method for a speech act theory-based semantic description of a language of communication acts.

1 Introduction

This research is concerned with communication between software agents. We see software agents as a paradigm that suggests a new way to view existing technologies as tools to build software applications that dynamically interact and communicate with their immediate environment (user, local resources and computer system) and/or the world, in an autonomous (or semi-autonomous), task-oriented fashion.

A crucial component of this paradigm is the communication language, which is the medium through which the attitudes regarding the content of an exchange between software agents are communicated; the communication language suggests whether the content of the communication is an assertion, a request, some form of query *etc.* Knowledge Query and Manipulation Language (KQML) is an agent communication language that consists of primitives (called *performatives*) which allow agents to communicate such attitudes to other agents and find other agents suitable to process their requests. Our research provides semantics for KQML along with a framework for the semantic description of KQML-like languages for agent communication. We do so, avoiding commitments to agent models and inter-agent interaction protocols.

2 KQML and the Problem of its Semantics

This is an example of a KQML message:

```
(ask-if :sender   A   :receiver    B   :language  prolog
        :ontology foo :reply-with  id1 :content   ``bar(a,b)'' )
```

In KQML terminology, *ask-if* is a *performative*. The value of the `:content` is an expression in some language (in this case in Prolog) or another KQML message and represents the content of the communication (illocutionary) act. The other parameters (*keywords*) introduce values that provide a context for the interpretation of the `:content` and hold information to facilitate the processing of the message. There is no such thing as an *implementation* of KQML, in the sense that KQML is not an *interpreted* or *compiled* language that is offered in some hardware platform or an abstract machine. Agents *speak* KQML because they use those primitives, this *library of communication acts*, with their reserved *meaning*. The application programmer is expected to provide code that processes each one of the performatives for the agent's language or knowledge representation framework.

KQML semantics have not been formally defined. Our goal is to provide a semantic description for the language, in a way that captures all the intuitions about the language, expressed in its existing documentation [1] without making commitments to specific agent models and coordination protocols in order to ensure the widest possible applicability of the language. There is good reason to supplement KQML with formal semantics. The lack of semantics for KQML has often been a source of criticism for KQML. Also, although various agent systems implementations that use KQML have appeared (such as the one described in the chapter entitled *"Facilitating Open Communication in Agent Systems: the Infosleuth Infrastructure"*, in this volume), there seems to be neither an agreement regarding the exact meaning of the used performatives nor a framework for defining the meaning of new performatives; these are problems that our semantic approach addresses. Moreover, agents can use the semantic definitions of performatives in order to make inferences resulting from the use of the KQML communication primitives. The semantic approach we propose uses expressions, that suggest the minimum set of preconditions and postconditions that govern the use of a performative, along with conditions that suggest the final state for the successful performance of the speech act (performative); these expressions describe the relevant to the exchange agents' states and use propositional attitudes like *belief*, *knowledge*, *desire*, *etc.* (this *intentional description* of an agent is only intended as a way of viewing the agent).

3 A Framework for the Semantics of Performatives

3.1 What Constitutes the Semantic Description

The following constitutes the semantic description for each of the performatives: **(1)** A natural language description of the performative's intuitive meaning; **(2)** An expression that describes the content of the communication act and serves as a formalization of the natural language description; **(3)** Preconditions that indicate what can be assumed to be (in part) the state of an agent when it sends a performative (**Pre(A)**) and what should be the state of the receiver in order to accept it and successfully process it (**Pre(B)**); **(4)** Postconditions that describe what can be assumed to be the (relevant to this message exchange) states of both interlocutors after the *successful* utterance of a performative (by the sender) and after the receipt and processing (but before a counter utterance) of a message (by the receiver). The postconditions (**Post(A)** and **Post(B)**, respectively) hold unless a *sorry* or an *error* is sent as a *response* in order to suggest the unsuccessful processing of the message; **(5)** A completion condition for the performative (**Completion**) that indicates the final state, after possibly a conversation has taken place and the intention suggested by the performative that started the conversation, has been fulfilled;

and **(6)** Any comments that we might find suitable to enhance the understanding of the performative.

3.2 Describing Agents' States

We use expressions in a meta-language to formally define (cognitive) states for agents and use them to describe the performative, the preconditions, postconditions and completion conditions associated with the use of a particular performative. In these expressions we use operators that stand for propositional attitudes and have a reserved meaning: **(1)** BEL, as in BEL(A,P), which has the meaning that P is (or can be proven) true for A; P is an expression in the native language of agent A; **(2)** KNOW, as in KNOW(A,S), expresses knowledge for S, where S is a state description (the same holds for the following two operators); **(3)** WANT, as in WANT(A,S), to mean that agent A desires the cognitive state (or action) described by S, to occur in the future; and **(4)** INT, as in INT(A,S), to mean that A has every intention of doing S and thus is committed to a course of action towards achieving S in the future. We also introduce two instances of actions: **(1)** PROC(A,M) refers to the action of A processing the KQML message M, meaning that the *received* and valid KQML message M is handled by the piece of code designated with processing the performative for the application (PROC(A,M) guarantees neither the proper processing of the message nor the conformance of the code with the semantic description); and **(2)** SENDMSG(A,B,M) refers to the action of A sending the KQML message **M** to B.

The argument of BEL is an expression P in the agent's implementation language. BEL(A,P) if and only if P is true for agent A; we do not assume any axioms for BEL. Roughly, KNOW, WANT and INT stand for the psychological states of knowledge, desire and intention, respectively. All three take an agent's state description (either a cognitive state or an action) as their arguments. An agent can KNOW an expression that refers to the agent's own state or some other agent's state description if it has been communicated to it. So, KNOW(A,BEL(A,"foo(a,b))) is a valid agent's state, as is KNOW(A,BEL(B,"foo(a,b)")), if BEL(B,"foo(a,b)") has been communicated to A with some message, but KNOW(A,"foo(a,b)") is not valid because "foo(A,B)" stands for an expression in the agent's knowledge store and not for a state description. Researchers have grappled for years with the problem of formally capturing the notions of *desire* and *intention* (the chapter entitled *"Intentional Agents and Goal Formation"*, in this volume, is just one such example). Various formalizations exist but none is considered a definitive one. We do not adopt a particular one neither we offer a formalization of our own. It is our belief that any of the existing formalizations would accommodate the modest use of WANT and INT in our framework.

3.3 A Language and Notation for Agents' States

For a KQML message **performative(A,B,X)**, **A** is the `:sender`, **B** is the `:receiver` and **X** is the `:content` of the performative (KQML message). We will use capital-case letters from the beginning of the alphabet (*e.g.*, *A*, *B*, *etc.*) for agents' names and letters towards the end of the alphabet (*e.g.*, *X*,*Y*,*Z*) for propositional contents of performatives. We also use *S* to refer to an agent's state and **M** for an instance of a KQML message.

All expressions in our language denote agents' states. Agents' states are either actions that have occurred (PROC and SENDMSG) or agents' mental states (BEL, KNOW, WANT or INT). We allow conjunctions ($\wedge$) and disjunctions ($\vee$) of expressions that stand for agents' states (the resulting expressions represent agents' states, also), but we do not allow $\wedge$ and $\vee$ in the scope of KNOW, WANT and INT. Propositions in the agent's native language can only appear in the scope of BEL and BEL can only take such a proposition as its argument. BEL, KNOW, WANT, INT and actions can be used as arguments for KNOW (actions should then be interpreted as actions that have already happened). WANT and INT can only use KNOW or an action as arguments. When actions are arguments of WANT or INT, they are actions to take place in the future.

A negation of a mental state is taken to mean that the mental state does not hold in the sense that it should not be inferred (we will use the symbol not). When $\neg$ qualifies BEL, *e.g.*, $\neg$ (BEL(A,X)), it is taken to mean that the `:content` expression X is not true for agent A, *i.e.*, it is not provable in A's knowledge base. Obviously, what "not provable" means is going to depend on the details of the particular agent system, for which we want to make no assumptions.

4 Semantics for three KQML Performatives

We present the semantics for three KQML performatives (*ask-if*, *tell* and *sorry*) in order to illustrate our approach. [1]

- **ask-if(A,B,X)**
  1. A wants to know what B believes regarding the truth status of the content X.
  2. WANT(A,KNOW(A,S))
     where S may be any of BEL(B,X), or $\neg$(BEL(B,X)).
  3. **Pre(A)**: WANT(A,KNOW(A,S)) $\wedge$ KNOW(A,INT(B,PROC(B,M)))
     where M is **ask-if(A,B,X)**
     Pre(B): INT(B,PROC(B,M))
  4. **Post(A)**: INT(A,KNOW(A,S))
     Post(B): KNOW(B,WANT(A,KNOW(A,S)))
  5. **Completion**: KNOW(A,S'))
     where S' is either BEL(B,X) or $\neg$(BEL(B,X)), but not necessarily the same instantiation of S that appears in $Post(A)$, for example.
  6. Not believing something is not necessarily the same with believing its negation (assuming that the language of B provides logical negation), although this may be the case for certain systems. The **Pre(A)** and **Pre(B)** suggest that a proper advertisement is needed to establish them.
- **tell(A,B,X)**
  1. A states to B that A believes the content to be true.
  2. BEL(A,X)
  3. **Pre(A)**: BEL(A,X) $\wedge$ KNOW(A,WANT(B,KNOW(B,S)))
     Pre(B): INT(B,KNOW(B,S))
     where S may be any of BEL(B,X), or $\neg$(BEL(B,X)).

[1] A more detailed account can be found in [3] and the semantics for the complete set in [2].

4. **Post(A)**: KNOW(A,KNOW(B,BEL(A,X)))
   Post(B): KNOW(B,BEL(A,X))
5. **Completion**: KNOW(B,BEL(A,X))
6. The completion condition holds, unless a *sorry* or *error* suggests B's inability to acknowledge the *tell* properly, as is the case with any other performative.

– **sorry(A,B,Id)**
1. A states to B that although it processed the message, it has no response to provide to the KQML message M identified by the `:reply-with` value **Id** (some message identifier).
2. PROC(A,M)
3. **Pre(A)**: PROC(A,M)
   Pre(B): SENDMSG(B,A,M)
4. **Post(A)**: KNOW(A,KNOW(B,PROC(A,M))) $\wedge$ $\mathsf{not}(Post_M(A))$,
   where $Post_M(A)$ is the **Post(A)** for message M.
   Post(B): KNOW(B,PROC(A,M)) $\wedge$ $\mathsf{not}(Post_M(B))$
5. **Completion**: KNOW(B,PROC(A,M))
6. The postconditions for M, as a result of message M do not hold. The not should be taken to mean that the mental state it qualifies should not be inferred to be true as a *result* of this particular message. This does not mean that for example $Post_M(B)$ does not hold if it has already been established by a previous message; it is up to B to decide (perhaps after using additional information) if and how it wants to alter its internal state with respect to the *sorry*.

5 Conclusions

KQML is a language for agent communication whose semantics have not been specified thus far. First attempts have been made but no complete semantic description for the full set of KQML performatives has appeared yet in the literature. We have devised a semantic framework for the semantic description of KQML-like languages, *i.e.*, languages of attitude-expressing communication primitives, for the communication between software agents. Our semantic framework separates the communication language from the agent model and the coordination protocol. We have used our approach to provide semantics for the full set of KQML primitives and we have presented the framework and the semantic description for three performatives.

References

1. ARPA Knowledge Sharing Initiative. Specification of the KQML agent-communication language. ARPA Knowledge Sharing Initiative, External Interfaces Working Group, July 1993.
2. Yannis Labrou. *Semantics for an Agent Communication Language*. PhD thesis, University of Maryland, Baltimore County, August 1996.
3. Yannis Labrou and Timothy Finin. Semantics and conversations for an agent communication language. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya, Japan, August 1997.