

Semantics for an Agent Communication Language

Yannis Labrou (contact) and **Tim Finin**
Computer Science and Electrical Engineering Department,
University of Maryland, Baltimore County,
1000 Hilltop Circle,
Baltimore, MD 21250, USA
Phone: (410) 455 2667
Fax: (410) 455 3969
{jklabrou,finin}@cs.umbc.edu

Semantics for an Agent Communication Language ^{*}

Yannis Labrou 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 (intelligent) software agents. Based on ideas from speech act theory, we present a semantic description for KQML that associates “cognitive” 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. Languages of communication acts address the issue of communication between software applications at a level of abstraction that could prove particularly useful to the emerging *software agents* paradigm of software design and development.

1 Introduction

This research is concerned with communication between software agents [13]. 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.

Agent technologies, *e.g.*, programming languages, communication languages, protocols and agent theories, may allow us to design and build software agents, but the essential property for the success of such applications in today’s paradigm of networked computing is their ability to effectively communicate and exchange *knowledge* with one another despite differences in hardware platforms, operating systems, architectures and programming languages. A crucial part of the solution to this *interoperability* problem is the communication language, which is the

^{*} This work was supported in part by the Air Force Office of Scientific Research under contract F49620-92-J-0174, and the Advanced Research Projects Agency monitored under USAF contracts F30602-93-C-0177 and F30602-93-C-0028 by Rome Laboratory.

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.

Although KQML's primitives are intuitively thought of as *speech acts*, we do *not* intend to provide a paradigm for the linguistic communication between software agents, a theory of *agency*, or a scheme for the semantic description of speech acts in general. We start with an introduction to KQML and the challenges of its semantic description, we follow with a coverage of the *speech act theory* that is relevant to our work, we then describe our semantic framework and give the semantics for a small set of KQML performatives (we have used our approach to describe the full set of KQML performatives). Next, we discuss our semantic approach with respect to similarly-intended research.

2 The Agent Communication Language KQML

KQML is a language for the exchange of information and knowledge between software agents, through message types that express an *attitude* regarding the actual expression being exchanged. This is 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*.¹ The value of the `:content` is an expression in some language 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. In this example, *A* is querying *B* (these are symbolic names for agents²), in *Prolog* (the `:language`), about the truth status of *bar(a,b)*. Any response to this KQML

¹ The term was first coined by Austin [3], to suggest that some verbs can be uttered so that they perform some action.

² We will use the term *agents* to indiscriminately refer to all kinds of KQML-speaking programs and applications.

message will be identified by *id1* (the `:reply-with`). The ontology³ named *foo* may provide additional information for the interpretation of the `:content`.

In an environment of KQML-speaking agents there are specialized agents called *facilitators* (*mediators* or *brokers* denote similarly intended agents [8]) to whom agents *advertise* their services and ask for assistance in finding other agents that can provide services for them. KQML is an abstraction, a collection of *communication* primitives with the assumptions of a simple model for inter-agent communication and an abstract design for KQML-speaking agents. There is no such thing as an *implementation* of KQML, *per se*, meaning that KQML is not an *interpreted* or *compiled* language that is offered in some hardware platform or an abstract machine. Agents *speak* KQML in the sense that 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.

2.1 The problem of semantics for KQML

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 [10,2] and is faithful to the linguistic origins of the language, 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 implementations have appeared, there seems to be neither an agreement regarding the exact meaning of the used performatives nor a framework for defining the meaning 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. Finally, the semantics of performatives can be used to devise specifications of interactions between KQML-speaking agents (conversations) [12].

We treat KQML performatives as speech acts. We adopt the descriptive framework for speech acts and particularly illocutionary acts suggested by Searle [16,15]. 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 An ontology is a repository of semantic and primarily pragmatic knowledge over a certain domain.

3 Speech Act Theory

Speech act theory is a high-level theoretical framework developed by philosophers and linguists to account for human communication. It has been extensively used, formalized and extended within the fields of Computational Linguistics and AI as a general model of communication between arbitrary agents. Austin pointed out that the major role of language, communication, is a kind of action [3]. He suggested that speakers do not simply utter sentences that are true or false, but rather perform speech actions such as requests, suggestions, promises, *etc.* As a result, speech act theory is concerned with language as action and speech acts are considered the minimal units of human communication. All utterances are speech acts, with the primary understanding that all utterances are *actions* of some sort.

The following three distinct actions can be identified in a speech act: (1) a *locution*, *i.e.*, the actual physical utterance (with a certain context and reference), (2) an *illocution*, *i.e.*, the conveying of the speakers' intentions to the listener, and (3) the *perlocutions*, *i.e.*, actions that occur as a result of the illocution. For example, "I order you to shut the door" is a locution, its utterance is the illocution of a command to shut the door and the perlocution may be (if all goes well) that the listener shuts the door. Most elementary speech acts that are performed by successful utterances are of the form $F(P)$ where F is the illocutionary force and P is the propositional content of the illocutionary act. The illocutionary force classifies speech acts to the following classes (variations of this classification also appear in the literature): *assertives*, *directives*, *commissives*, *declaratives* and *expressives*.

Searle, in extending Austin's ideas, tried to specify necessary and sufficient conditions for the successful performance of speech acts, involving the intentions of the participants [16]. In [15], he introduced a formalization of illocutionary acts, and analyzed the illocutionary force into seven components, attempting to capture all aspects of the illocutionary act. The illocutionary force should be regarded as the sum of these seven components, namely: the *illocutionary point*, the *degree of strength* of the illocutionary point, the *mode of achievement*, the *propositional content conditions*, the *preparatory conditions*, the *sincerity conditions*, and finally, the *degree of strength of the sincerity conditions*.

4 A Framework for the Semantics of KQML Performatives

4.1 Performatives as speech acts

Instead of providing a comprehensive definition of agency, we suggest that agents are commonly taken to be "high-level" (*i.e.*, they use symbolic representation, display cognitive-like function and are commonly viewed as having an intentional level description (*i.e.*, their state is viewed as consisting of mental components such as beliefs, capabilities, desires, choices, commitments, *etc.*). We take this

intentional description [9] to be a helpful *abstract model* for *viewing* software agents, even if their actual implementation does not make such claims.

We next describe how we relate speech act-oriented accounts to the states of the intentionally described agents that use the performatives. We treat the *propositional content conditions as preconditions* on the listener. The *preparatory conditions*, for an agent to use a performative, are viewed as *preconditions* on the state of the speaker. For the *perlocutionary effects* we provide the state of the receiver, after processing a message (a *postcondition*). We also provide the state of the sender, after sending a message (another *postcondition*). The objective is to help with the interpretation of the performative, by suggesting the desired effects of its use (on both the sender and the receiver). Finally, we need to know when the illocutionary point of a performative is eventually satisfied, *e.g.*, a query is satisfied when it is answered appropriately. Other illocutionary acts are satisfied just by being uttered, such as telling (*tell*), and others, like asking (*ask-if*), require a further exchange of messages, *i.e.*, a *conversation*. Thus, we provide a *completion* condition, which indicates the state of affairs *after* the completion of the illocutionary act (performative). Preconditions, postconditions and completion conditions describe states of the involved agents.

4.2 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 illocutionary act. For all practical purposes, this is a formalization of the natural language description.
3. Preconditions that indicate the necessary state for an agent in order 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 a *error* or *sorry* will be the most likely response.
4. Postconditions that describe the 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.
6. Any comments that we might find suitable to enhance the understanding of the performative.

4.3 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 (speech act). In these expressions we use operators that stand for propositional attitudes and have a (informal) reserved meaning. The operators we use are:

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 ⁴.
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.
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 . Every message after being *received* is *processed*, in the sense that it is a valid KQML message and the piece of code designated with processing the performative for the application indeed processes it. PROC(A,M) does not guarantee proper processing of the message (or conformance of the code with the semantic description).
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 (in the *model-theoretic* sense) 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 valid, 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*. 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.

⁴ The *native* language of the application may or may not have modal operators but in our analysis we do not assume any.

4.4 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). From a *speech act* point of view, **A** is the speaker, **B** is the listener and **X** is the propositional content of the speech act that takes place. Occasionally we use **M** to refer to an instance of a 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. All underscores ($_$) are unnamed, universally quantified variables (they stand for performative parameters that do not have values in the KQML message). We will use capital case letters preceded by a question mark ($_?$), *e.g.*, $_?B$, for existentially quantified variables.

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 **not** qualifies **BEL**, *e.g.*, **not**(**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.

5 Semantics for KQML Performatives

We present the semantics for five KQML performatives (*advertise*, *ask-if*, *tell*, *sorry* and *broker-one*) which can support some interesting agent conversations and illustrate our approach.⁵

– **advertise(A,B,M)**

1. **A** states to **B** that **A** can and will process the message **M** from **B**, if it receives one (**A** commits itself to such a course of action).
2. **INT**(**A**,**PROC**(**A**,**M**))
where **M** is the KQML message **performative_name(B,A,X)**.
3. **Pre(A)**: **INT**(**A**,**PROC**(**A**,**M**))
Pre(B): **NONE**

⁵ The semantics for the complete set are given in [11].

4. **Post(A)**: $\text{KNOW}(A, \text{KNOW}(B, \text{INT}(A, \text{PROC}(A, M))))$
Post(B): $\text{KNOW}(B, \text{INT}(A, \text{PROC}(A, M)))$
 5. **Completion**: $\text{KNOW}(B, \text{INT}(A, \text{PROC}(A, M)))$
 6. An *advertise* is a commissive act, in the sense that it commits its sender to process M , as suggested by the announcement of the intention to process. If B is a *facilitator* then B is interchangeable (in the semantic description) with the name of any agent the facilitator knows about.
- **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 $\text{BEL}(B, X)$, or $\neg(\text{BEL}(B, X))$.
 3. **Pre(A)**: $\text{WANT}(A, \text{KNOW}(A, S)) \wedge \text{KNOW}(A, \text{INT}(B, \text{PROC}(B, M)))$
where M is **ask-if(A,B,X)**
Pre(B): $\text{INT}(B, \text{PROC}(B, M))$
 4. **Post(A)**: $\text{INT}(A, \text{KNOW}(A, S))$
Post(B): $\text{KNOW}(B, \text{WANT}(A, \text{KNOW}(A, S)))$
 5. **Completion**: $\text{KNOW}(A, S')$
where S' is either $\text{BEL}(B, X)$ or $\neg(\text{BEL}(B, X))$, but not necessarily the same instantiation of S that appears in $\text{Post}(A)$, 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 (see *advertise* and our comments in Section 6).
- **tell(A,B,X)**
1. A states to B that A believes the content to be true.
 2. $\text{BEL}(A, X)$
 3. **Pre(A)**: $\text{BEL}(A, X) \wedge \text{KNOW}(A, \text{WANT}(B, \text{KNOW}(B, S)))$
Pre(B): $\text{INT}(B, \text{KNOW}(B, S))$
where S may be any of $\text{BEL}(B, X)$, or $\neg(\text{BEL}(B, X))$.
 4. **Post(A)**: $\text{KNOW}(A, \text{KNOW}(B, \text{BEL}(A, X)))$
Post(B): $\text{KNOW}(B, \text{BEL}(A, X))$
 5. **Completion**: $\text{KNOW}(B, \text{BEL}(A, X))$
 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 (possibly further) response to provide to the KQML message M identified by the `:reply-with` value **Id** (some message identifier).
 2. $\text{PROC}(A, M)$
 3. **Pre(A)**: $\text{PROC}(A, M)$
Pre(B): $\text{SENDMSG}(B, A, M)$
 4. **Post(A)**: $\text{KNOW}(A, \text{KNOW}(B, \text{PROC}(A, M))) \wedge \text{not}(\text{Post}_M(A))$,
where $\text{Post}_M(A)$ is the **Post(A)** for message M .
Post(B): $\text{KNOW}(B, \text{PROC}(A, M)) \wedge \text{not}(\text{Post}_M(B))$

5. **Completion:** KNOW(B,PROC(A,M))
 6. The postconditions and completion conditions do not hold, even though A dispatched the performative to the appropriate function, because A could not (or did not want) to come up with a response that would result to their satisfiability. 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*.
- **broker-one(A,B,performative(A,-,X))**

Let D be an agent such that $CANPROC(D,performative(B,D,X))$ ⁶ and *performative* be a performative that entails a request (a *directive*); for the set of performatives presented here, only *ask-if* falls into this category. Then B sends **performative(B,D,X)** to D , receives some *response* (depending on the *performative*) from D , let us call it **response(D,B,X')**, and then B sends to A the message **forward(B,A,-,A,response(-,A,X'))**.⁷

Semantically this is a three-party situation. We break down the semantic description to the three (agent) pairs involved in the transaction.

A and B For A and B , the semantics are **not** those of a **performative(A,-,B,X)**, meaning that A is aware that whatever *response*, if any, comes from B is merely an “echo” of the utterance of the broker-ed agent D .

- **broker-one(A,B,ask-if(A,-,X))** (for A and the broker B)
 1. A wants B (a broker) to send the **:content** of the *broker-one* to some agent that can process it and eventually forward the response of the broker-ed agent back to A .
 2. **WANT(A,SENDMSG(B,D,M))**
where M is **performative(B,D,X)** and D is an agent such that $CANPROC(D,M)$
 3. **Pre(A): WANT(A,SENDMSG(B,D,M))**
Pre(B): B has to be a *facilitator*; an agent can be a facilitator if and only if it can process performatives like *broker-one*, although it is usually more helpful to ascribe facilitator status to an agent in advance, so that agents can know which agent to contact for such requests.

⁶ $CANPROC$, as in $CANPROC(A,M)$, stands for “ A being able to process message M .” It is always the case that if **advertise(A,B,M)** then $CANPROC(A,M)$, but it could very well be the case that $CANPROC(A,M)$ may be inferred in other ways (this is to be provided or inferred by B). $CANPROC$ is entirely different from $PROC$; $CANPROC$ suggests ability to process and $PROC$ suggest that the agent will process (or has already processed) a performative, in the sense that it will (or did) dispatch the message to the appropriate piece of code for handling.

⁷ The performative *forward* is not presented here. Its meaning is basically the intuitive one and the four parameters **:from**, **:to**, **:sender** and **:receiver** refer respectively to the originator of the performative in the **:content**, the final destination, the **:sender** of the *forward* and the **:receiver** of the *forward*.

4. **Post(A)**: KNOW(A,SENDMSG(B,D,M))
Post(B): SENDMSG(B,D,M)
5. **Completion**: SENDMSG(B,A,forward(B,A,-,A,M'))
 where M' is the message **response(-,A,X')** generated by the brokered agent's response to B, *i.e.*, **response(D,B,X')**.
6. To offer an example, if the **:content** of the *broker-one* was **ask-if(A,-,X)**, A understands that the (possible) *response forward(B,-A,-,A,tell(-,A,X))* does not imply that BEL(B,X), since D's response to B is wrapped in a *forward* and then sent to A. Also, D's name is omitted in the *forward*, so A does not know D's name.

B and D For B and D the semantics are those of **performative(B,D,X)**, meaning that as far as D knows of, the exchange has the meaning and repercussions of **performative(B,D,X)** (and whatever additional responses) being exchanged between B and D.

A and D For A and D the semantics are those of **performative(A,D,-X)** (let us call it M) but with the major difference that this is an one-sided exchange. So, $Pre_M(D)$ and $Post_M(D)$ are empty because D does not know that it has this exchange with A. Additionally, A can have no prior knowledge (in $Pre_M(A)$) of its interlocutor's state. Finally, the applicable $Post_M(A)$ and $Completion_M$ lack the name of D; for example, in the case of the exchange mentioned above (a *broker-one* with **:content ask-if(A,-,X)**), the $Completion_M$ will eventually be KNOW(A,BEL(?D,X)) meaning A knows that there is some agent that BEL X but A does not know the identity (name) of the agent, even though B does. The following is the semantics of **broker-one(A,B,ask-if(A,-,X))** for agent A and the brokered agent D.

- **broker-one(A,B,ask-if(A,-,X))** (for A and the brokered agent D)
 1. A wants to know what some other agent believes regarding the truth status of the content X.
 2. WANT(A,KNOW(A,S))
 where S may be any of BEL(?D,X), or \neg (BEL(?D,X)).
 3. **Pre(A)**: WANT(A,KNOW(A,S))
Pre(D): NONE
 4. **Post(A)**: INT(A,KNOW(A,S))
Post(D): NONE
 5. **Completion**: KNOW(A,S'))
 where S' is either BEL(?D,X) or \neg (BEL(?D,X)), but not necessarily the same instantiation of S that appears in $Post(A)$, for example.
 6. In effect, D's identity remains unknown to A and D is unaware that A knows its belief regarding the truth status of X.

6 Discussion

Ever since [4] speech acts have been thought as planning operators in the planning tradition of AI. Cohen [7] introduced a plan-based treatment of speech acts to which our approach bears similarities to. Appelt's work [1] supplements

this tradition with *possible-worlds* semantics for the modalities. In more recent work, Cohen and Lesveque [5] suggest a model for rational agents, which uses a *possible-worlds* formalism, that can in turn be used as a framework for the semantic description of illocutionary acts [6]. Sadek [14] has also taken on a similar task of defining rational agency and defining communicative acts on top of it. Finally, Singh proposes a model of agency [17], which differs from that of Cohen and uses it as a framework for the semantic treatment of speech acts [18]. In contrast, we draw directly from a high-level speech act account, although the resulting preconditions/postconditions framework is reminiscent of planning (but it could also be thought as operational semantics, *i.e.*, transitions on agents' states).

In our approach we provide no formal semantics (in a *possible-worlds* formalism or some similar framework) for the modal operators but we restrict the scope and use of these operators, so that they can be subsumed by similar modalities whose semantics could be provided by an intentional theory of agency. The complexity of possible-worlds-like formalisms can be prohibiting for the intended audience of our semantic description that includes application developers that want to support KQML in their software agents. We do not expect such users of KQML to be fluent in modal logic(s) or KQML-speaking agents to actually implement complex agent theories. Alternatively, of course, KQML developers could use the semantic definition only as a reference model and the (never to be tested) conformance will be established by virtue of declaring the adoption of the reference model. But the larger problem remains, namely, which theory of agency should be used as the substrate for the semantics of the communication primitives? Different researchers have provided different accounts (such as [6], [19], [17] or [14], for example). Another common element of the mentioned approaches is the strictly declarative definitions of the primitives. Instead, our preconditions, postconditions and completion conditions framework suggests a more operational way of thinking which we hope will be useful to implementors that have to provide the code that processes the communication primitives.

By attempting a semantics for communication acts without a theory of agency, *i.e.*, formal semantics for the propositional attitudes (operators), we certainly give up interesting inferencing. For example, if an agent sends $\mathbf{tell(A,B,X)}$ and later $\mathbf{tell(A,B,X \rightarrow Y)}$, B will not be able to infer that $\mathbf{BEL(A,Y)}$ (since we do not even assume a universal *weak S4* model for \mathbf{BEL}) based on the KQML semantics alone. But, if B has additional information about A such information may be inferred. Similar observations can be made about the other modalities. In the end, we trade a formal semantics for the propositional attitudes, which inevitably define a *model of agency* that is unlikely to be universal for all agents, for a simpler formalism and agent theory independence. Nothing is lost though, because the additional information of the agent theory that holds for the agent can be supplied as part of the KQML exchange (*e.g.*, in the `:ontology` value of a KQML message) and subsequently taken into consideration for further inferencing.

Objections may be raised regarding some of our choices regarding the meaning we chose to attribute to some of the performatives. Our semantics for *tell*, for example, suggest that an agent can not offer unsolicited information to some other agent. This can be easily amended by introducing another performative, let us call it *proactive-tell* which has the same semantic description as *tell* with the following difference: **Pre(A)** is $\text{BEL}(A,X)$, and **Pre(B)** is empty. Similarly, an agent *A* can send an *ask-if* to agent *B* if and only if *A* knows that *B* is going to process such a request. Implicit in this choice, is our preference for a model where agents advertise their services so that other agents (with the help of *mediators* or *facilitators*) can find agents that can process requests for them. A “relaxed” version of *ask-if* can be introduced to allow for direct querying. The semantic description of this *proactive-ask-if* differs from that of *ask-if* as follows: **Pre(A)** is $\text{WANT}(A,\text{KNOW}(A,S))$, and **Pre(B)** is empty. In the end, following KQML’s tradition of an open standard, the KQML users’ community should decide the performative names to be associated with whatever semantic description. Finally, it might also be useful to introduce a performative that will allow an agent to directly inquire another agent about its willingness to process a particular request.

7 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, modeled after speech acts, for the linguistic 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 a handful of performatives.

KQML-like languages for agent communication address the issues of knowledge exchange through a layered approach, by separating issues of representation from issues of communication. The communication primitives are opaque to the content they carry and only express attitudes about it. Translation between languages and/or knowledge representation schemes and means to share the ontological commitments of software agents will further support KQML-like languages as a medium for knowledge sharing between software agents.

References

1. Douglas E. Appelt. *Planning English Sentences*. Cambridge University Press, Cambridge, UK, 1985.
2. ARPA Knowledge Sharing Initiative. Specification of the KQML agent-communication language. ARPA Knowledge Sharing Initiative, External Interfaces Working Group working paper., July 1993.

3. J.L. Austin. *How to do things with words*. Harvard University Press, Cambridge, MA, 1962.
4. B. C. Bruce and C .F. Schmidt. Belief systems and language understanding. Technical Report No. 2973, Bolt Beranek and Newman Inc., Cambridge, Mass., January 1975.
5. Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
6. Philip R. Cohen and H.J. Levesque. Communicative actions for artificial agents. In *Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS'95)*. AAAI Press, June 1995.
7. P.R. Cohen and C.R. Perrault. Elements of a plan-based theory of speech acts (1979). In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 169–186. Morgan Kaufman Publishers, San Mateo, CA, 1988.
8. Keith Decker, Mike Williamson, and Katia Sycara. Matchmaking and brokering. In *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS'96)*, December 1996.
9. Daniel C. Dennett. *BRAINSTORMS: Philosophical Essays on Mind and Psychology*. Bradford Books, Publishers, Cambridge, 1978.
10. T. Finin, R. Fritzson, and D. McKay et. al. An overview of KQML: A knowledge query and manipulation language. Technical report, Department of Computer Science, University of Maryland Baltimore County, 1992.
11. Yannis Labrou. *Semantics for an Agent Communication Language*. PhD thesis, University of Maryland, Baltimore County, August 1996.
12. 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. (to appear).
13. Charles Petrie. Agent-based engineering, the web, and intelligence. *IEEE Expert*, December 1996.
14. M.D. Sadek. A study in the logic of intention. In *Proceedings of the 3rd Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 462–473, Cambridge, MA, 1992.
15. J. Searle and D. Vanderveken. *Foundations of illocutionary logic*. Cambridge University Press, Cambridge, UK, 1985.
16. John R. Searle. *Speech Acts*. Cambridge University Press, Cambridge, UK, 1969.
17. M.P. Singh. A logic of intentions and beliefs. *Journal of Philosophical Logic*, 22:513–544, 1993.
18. M.P. Singh. A semantics for speech acts. *Annals of Mathematics and Artificial Intelligence*, 8(I-II):47–71, 1993.
19. Ira A. Smith and Philip R. Cohen. Toward a semantics for an agent communications language based on speech-acts. In *Proceedings of the 13th National Conference on Artificial Intelligence*. AAAI/MIT Press, August 1996.