# CG-KQML+: An Agent Communication Language and its use in a Multi-Agent System

Karim Bouzouba[1], Bernard Moulin[2], Adil Kabbaj[3]

[1] Abdelmalek Essaadi University
ENSAT, Tangier, Morocco
karimbouzouba@yahoo.com
[2] Laval University
Computer Science Department and Research Center on Geomatics
Québec, Québec, Canada G1K 7P4
moulin@ift.ulaval.ca
[3]INSEA, Rabat, Morocco, B.P. 6217
akabbaj@insea.ac.ma

**Abstract** This paper deals with communication protocols between agents and between agents and users [3]. It presents a new communication model which is based on a careful analysis of speech act theory and on two fundamental principles applied to communication: a) communication is considered as a negotiation process and, b) communication results in an exchange of mental states.

Using this model of communication and the conceptual graph formalism for the representational level, we propose a new agent communication language, called CG-KQML+ which is an extension of the KQML language.

The paper also shows the use of CG-KQML+ in a MAS called POSTAGE which aims at helping users in their correspondence task. In POSTAGE, software agents manage administrative correspondence on behalf of and in cooperation with their users. Users and agents have interactions which respect administrative correspondence rules. A POSTAGE agent is responsible for sending the generated message to the addressee's POSTAGE agent. The paper presents the second version of POSTAGE which is implemented using the Prolog+CG language.

## 1 Introduction

A Multi-Agent System (MAS) is composed of multiple heterogeneous intelligent software systems (called agents) which can compete, negotiate or cooperate [19]. In recent years the interest in MAS has grown tremendously and today agent technology is being used in a large range of industrial applications.

Agent technology is widely used to help users to achieve various tasks in diverse domains such as network management [23], air-traffic control [6], telecommunications [4], and electronic commerce [5]. Agent applications have one thing in common: agents must be able to communicate in order to decide which actions to perform. In a multi-agent system, we distinguish two levels of communication: agent/user communication and agent/agent communication. In both kinds of communication, industry is trying to find a standard communication language. To assist developers in the use of agent technology, some multi-agent system tools have been developed [17, 21]. However, these tools are still suffering from the following drawbacks: a) there is no standard architecture for MASs; b) there is no standard design method for the design of a MAS; c) there is no standard communication protocol agents /users.

This paper deals with communication protocols between agents and between agents and users [3]. It presents a new communication model which is based on a deep analysis of speech act theory [22] [28] and on two fundamental principles: a) communication is considered as a negotiation process [14, 18], b) communication results in an exchange of mental states [7, 24]. Thus, we consider agents' communication as exchanges of mental states (goals, beliefs, etc.) and exchanges of what we call communicational states (CS). Communication is considered as a negotiation game where agents negotiate about proposed CSs. An agent proposes a CS and other agents react to the proposal by accepting, rejecting the proposed CS or even asking for further information. Such an action establishes a relationship between the CS and the agent that is called an agent's positioning.

Using this model of communication and the conceptual graph formalism for the representational level, we developed a new agent communication language, called CG-KQML+ which is an extension of the KQML language [12]. CG-KQML+ overcomes some limitations of KQML: KQML performatives are limited to the assertive and directive categories, inappropriate choice of performatives, different interpretations of KQML performatives. The paper also shows the use of CG-KQML+ in a MAS called POSTAGE (POSTman AGEnt) [2]. The aim of this MAS is to help users to achieve correspondence tasks. In POSTAGE, software agents manage administrative correspondence on behalf of and in cooperation with their users. Users and agents interact respecting administrative correspondence rules. A POSTAGE agent is responsible for sending the generated message to the addressee's POSTAGE agent.

A first version of POSTAGE has been implemented using ECLIPSE [11] and Delphi [9]. Since that time and by using the Conceptual Graph formalism more fully, we enhanced our standardization work as well as our formulation of POSTAGE. Now, a new version of POSTAGE has been implemented with Prolog+CG language [15]. Being a CG-based extension of Prolog, Prolog+CG provides the abstraction level needed to easily implement a CG-based application. Indeed, our new version of POSTAGE is more concise and readable. Moreover, the integration of Java and Prolog+CG [16] enabled us to develop the front/end interface using Java and the kernel of the system using Prolog+CG.

Section 2 presents our agent communication model. Section 3 presents CG-KQML+. Section 4 presents the POSTAGE multi-agent system. Section 5 discusses some future works and concludes the paper.

## 2 The communication model

When interacting, agents can engage in two kinds of communication: agent/user communication and inter-agent communication (Figure 1). Agents communicate with users in order to characterize their needs and to provide them with answers or solutions. Agents communicate with each other in order to exchange various kinds of information. When communicating with other agents, an agent uses a specific Agent Communication Language (ACL). An agent's architecture contains a communication process which handles communication activities as well as other processes used to perform various tasks such as planning, decision making or negotiation. In this paper, we focus on the communication activity.

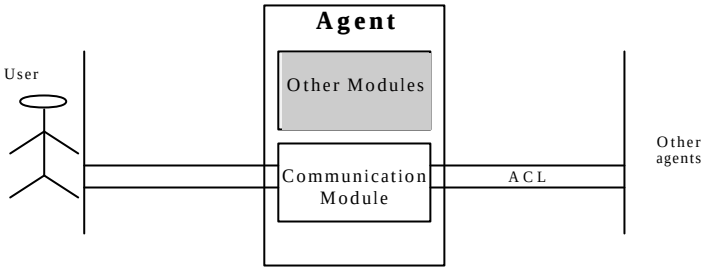


**Fig1**. Agent Architecture

Several researchers consider that agent communication should respect two fundamental principles: (1) Communication is considered as a negotiation process [14, 18]; (2) Communication results in an exchange of mental states [7, 24]. We adopt these principles in our approach and we consider agents' communication as exchanges of mental states (goals, beliefs, etc.) and exchanges of what we call communicational states[1] (CS). Communication is considered as a negotiation game where agents negotiate about CSs.

A CS is characterized by one of five types, each type corresponding to a performative type as defined by Vanderveken [28][2]. According to speech act theory, a speech act can be expressed using a performative verb. These verbs are grouped into five categories: a directive verb (e.g. order) allows one to formulate queries, an assertive verb (e.g. tell) allows one to state facts, a commissive verb (e.g. promise) can be used to commit to perform an action, an expressive verb (e.g. hate) allows one to express a mental attitude, and a declarative verb (e.g. declare) can be used to make declarations.

A directive CS is performed by an agent i toward an agent j at time t concerning the propositional content Prop. It has the following form[3] :

```
[CS : Id]-
       -hasType->[TYPE : directive],
       -hasSender->[AGENT : i],
       -hasReceiver->[AGENT : j],
       -timeOf->[TIME : t],
       -hasContent->[PROPOSITION : Prop].
```

The other types of CSs are assertive, declarative, commitment and expressive. We consider four kinds of agent's positioning relative to CSs: the proposition `propose`, the acceptance `accept` and the refusal `refuse` of a CS, as well as the `inquire` positioning to ask questions. A positioning is represented using the following generic CG:

```
[POSIT : Id]-
              -hasType->[TYPE : posit],
              -hasSender->[AGENT : i],
              -hasReceiver->[AGENT : j],
              -TimeOf->[TIME : t],
              -hasPower->[SOCIAL_POWER : power],
              -hasCS->[CONTENT : CS_CG].
```

[1] We choosed the term "Communicational State" analogously to the term "Mental State".

[2] Most of the terminology concerning the communication module and the ACL are borrowed from speech act theory [22] which is the field that analyses speech acts from philosophical and logical points of view.

[3] For the formalization of communicational states, we have been inspired by [10].

This CG represents the positioning of agent i toward agent j at time t with respect to a certain CS. In addition, i positions itself relative to a certain social power which can have the values of `peer` (for a peer power between i and j), `power` (i has power over j), and `not-power` (j has power over i). For example, let us take the following simple dialogue between agents A1 and A2 which are in a peer power position:

| | | |
|---|---|---|
| (SA1) | A1: | Print the document number 5 |
| (SA2) | A2: | Ok ! |

The speech act SA1 is represented by a proposition of a directive CS: A1 is proposing to A2, at time t1 and in the peer social power relation, a directive where A1 is asking A2 at time t1 that agent A2 print the object document number 5.

```
[POSIT : p1]-
   -hasType->[TYPE : propose],
   -hasSender->[AGENT : A1],
   -hasReceiver->[AGENT : A2],
   -timeOf->[Time : t1],
   -hasPower->[SOCIAL_POWER: peer],
   -hasCS->[CONTENT: [CS:cs1]-
                     -hasType->[TYPE : directive],
                     -hasSender->[AGENT : A1],
                     -hasReceiver->[AGENT : A2],
                     -timeOf->[Time : t1],
                     -hasContent->[PROPOSITION:
              [AGENT:A2]<-agnt-[PRINT]-obj->[DOCUMENT:document5] ], ].
```

The speech act SA2 is represented by the acceptance of the first directive: At time t2, A2 is accepting, in the peer social power relation, the directive made at time t1 by agent A1 which was asking A2 to print the document number 5.

```
[POSIT : p2]-
   -hasType->[TYPE : accept],
   -hasSender->[AGENT : A2],
   -hasReceiver->[AGENT : A1],
   -timeOf->[Time : t2],
   -hasPower->[SOCIAL_POWER: peer],
   -hasCS->[CONTENT:
              [CS:cs1]-
                     -hasType->[TYPE : directive],
                     -hasSender->[AGENT : A1],
                     -hasReceiver->[AGENT : A2],
                     -timeOf->[Time : t1],
                     -hasContent->[PROPOSITION:
              [AGENT:A2]<-agnt-[PRINT]-obj->[DOCUMENT:document5] ], ].
```

In our approach, each agent records the sequence of CSs and their associated positionings, exchanged during a communication, into a conceptual structure called the communicational trace. Using the communicational traces of both agents, the above dialogue can be represented by the following figure where CSs are represented in circles and positionings using arrows pointing towards the circles:

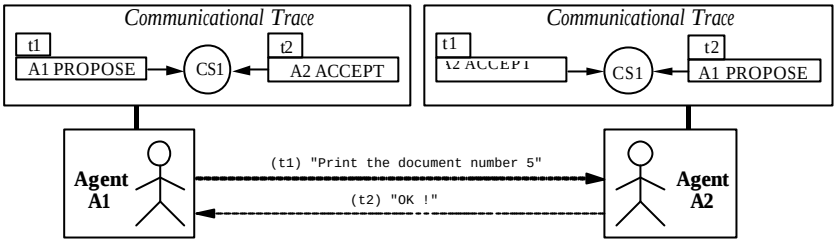


**Fig. 2.** Communicational traces of agents A1 and A2

In this section we briefly presented our communication model which allows an agent to reason about its communication activities. Let us examine now the main characteristics of the proposed ACL.

## 3 CG-KQLM+ : an extension of KQML

A first attempt to provide a standardized Agent Communication Language (ACL) came forth from the ARPA knowledge sharing project and produced KQML [12]. In the context of that project, researchers developed two main components: (1) a representation language used to express message contents (called Knowledge Interchange Format or KIF); and (2) a communication language KQML (Knowledge Query Manipulation Language). KQML has been designed to facilitate high level cooperation and interoperation among agents. KQML offers an extensible set of so-called performatives which specify what kind of communication actions agents can perform. KQML performatives are either assertive (used to state a fact) or directives (used to give orders or send requests). A typical KQML message has the following syntax :

```
(tell
      :sender        A
      :receiver      B
      :content       "raining")
```

The above KQML message means that agent A tells agent B that "raining" is true. There are other ACLs based on speech act theory such as FIPA [13], COOL [1] and AOP [25]. Our study of KQML has outlined the following limitations:

1. *KQML performatives are limited to the assertive and directive categories.* There is no performative allowing agents to perform declarative, expressive, or commissive speech acts. However, without a commissive performative, it is difficult to imagine how agents could cooperate. Indeed, whenever an agent is busy and cannot immediately execute a requested action, it should send a commissive message, promising the requesting agent that the answer will be forthcoming. Hence, without the commissive performative, the requesting agent would have no way to know if it will get an answer for its request.

2. *Inappropriate choice of performatives.* For example, when an agent cannot answer to a request, it uses the performative sorry. But, sorry is not a verb as should be a performative according to speech act theory. In addition, this performative displays a psychological attitude of regret. However, the agent may have no regret when

performing an act using this performative. More details about inappropriate choices of other KQML performatives can be found in [8].

3. *Different interpretations of KQML performatives*. Although KQML is supposed to be a standard language, several projects using KQML interpret differently certain performatives. For example, in order to announce its abilities, an InfoSleuth agent [20] uses the performative `tell` while a Visitor-Hoster agent [27] uses the performative `advertise`.

Because of these limitations, and basing our proposal on a rigorous study of speech act theory, we extended KQML by proposing new performatives and new slots in the structure of a message. The structure that we proposed for a message [3] can be described by the following CG:

```
[KQML+_MESSAGE]-
              -hasPrimPerf->[PERFORMATIVE],
              -hasExprPerf->[PERFORMATIVE],
              -hasExprSocP->[SOC: social_position],
              -hasPresentation->[presesentation],
              -hasSender->[AGENT : sender],
              -hasReceiver->[AGENT : receiver],
              -hasContent->[CONTENT : CG],
              -replyWith->[NUMBER : number],
              -inReplyTo->[NUMBER : number].
```

The primitive-performative verb (introduced by the relation hasPrimPerf) can be of five types and corresponds to one of the five primitive verbs suggested by speech act theory. For example, to perform a directive, the verb direct is the primitive performative verb suggested by speech act theory. The selected verbs of the other performative types are listed in Table 1:

**Table 1.** Primitive verbs for each performative type

| *Performative* | *Meaning* |
|---|---|
| direct | for a directive |
| assert | for an assertive |
| declare | for a declarative |
| express | for an expressive |
| commit | for a commissive |

The primitive-performative allows to overcome the multiple interpretations' problem and since we selected all the five types, we allow agents to exchange any kind of speech acts (we recall that in KQML, only directives and assertives are possible). In some cases, the primitive verb cannot be used. The verb announce for example, might be used to perform a declarative speech act instead of the verb declare. The expressed-performative (hasExprPerf) relation is added to the Message structure for that purpose. expressed-social-position (hasExprSocP) allows an agent to convey social relationships. When the message includes some "presentation packaging" which strengthen the speech act, we use the presentation relation (hasPresentation).

KQML uses the KIF format to express the message's propositional content. The Knowledge Interchange Format (KIF) is a language designed to enable agents exchange knowledge, but is not intended to provide an internal knowledge representation for agents. Typically, when an agent reads a knowledge base in KIF, it converts the data into its own internal form. When the agent needs to communicate with another agent, it maps its internal data structures into KIF [26]. Since the core of the internal representation of our agents is based on the conceptual graph notation and in order to avoid any conversion, we prefer to use the CG notation as the propositional content of a CG-KQML+ message. For example, the speech act "I ask you to send me the parcel", is represented by the following CG:

```
[KQML+_MESSAGE]-
       -hasSender->[AGENT : *1 = I],
       -hasReceiver->[AGENT : You],
       -hasPrimPerf->[DIRECT],
       -hasExprPerf->[ASK],
       -hasContent->[CONTENT: [SEND]-agnt->[AGENT:You]
                                -obj->[PARCEL]
                                -ptnt->[AGENT:*1 = me] ][4],
       -replyWith->[NUMBER : r1].
```

The following speech act "As a general manager, I deeply regret having to announce your dismissal from our company" performed by A1 is transmitted to A2 using the following CG-KQML+ message:

```
[KQML+_MESSAGE]-
   -hasSender->[AGENT : A1],
   -hasReceiver->[AGENT : A2],
   -hasPrimPerf->[DECLARE],
   -hasExprPerf->[ANNOUNCE],
   -hasExprSocP->[SOC: general_manager],
   -hasPresentation->[REGRET]-manr->[DEEP],
   -hasContent->[CONTENT:[AGENT:A1]<-agnt-[DISMISS]-ptnt->[AGENT: A2]],
   -replyWith->[NUMBER : r1].
```

## 4 The communication model in POSTAGE

The CG-formalism has been used for different types of applications such as natural language processing applications. To our knowledge the use of the CG formalism in a MAS has not been tested yet. We focus on the communication side of this type of application. The CG-formalism provides, among other advantages, the standardization of the communication protocol (which is a needed feature) inside a MAS.

We present here a second version of the POSTAGE application (the first version was not using the CG-formalism). POSTAGE is a multi-agent system which manages administrative correspondence on behalf of and in cooperation with users. In large and small organizations, such a correspondence exists in various forms: formal and informal letters, memos, notices, etc. The way those documents are written depends on the social relationships existing between the interlocutors. Messages may be very informal, very polite or quite informal.

Therefore, developing a software agent to which the administrative correspondence task is delegated would greatly benefit to the user: (1) the user is not obliged to remember all the

[4] The multi-referent "*1" is used to specify that the two concepts [AGENT : *1 = I] and [AGENT:*1 = me] are in fact two occurrences of the same concept.

formulations used in his/her organization, (2) the user can respect the social hierarchy of his/her organization.

Using POSTAGE, a user can formulate a message in an informal way, and the POSTAGE agent will transform this message in a way which agrees with (1) the social relationship existing between the user and his addressee; (2) the user communicative intention and (3) the formulation rules used in a particular organization. For example, the informal message "You are laid off" would be transformed into "As general manager, I deeply regret having to announce your dismissal from our company". For our present work, we choose the university organization as an example for the development of POSTAGE. A POSTAGE agent has a specific architecture which allows it to perform the correspondence task (Figure 3). This architecture is divided into two parts. The first part includes four knowledge models and the second one three execution modules. The user's model contains knowledge concerning the user such as his/her preferences and his/her social relationships with other users. The static knowledge contains plans and specific formulation schemas.

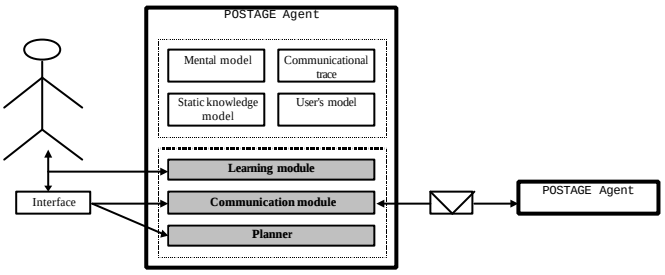


**Fig. 3.** Architecture of a POSTAGE agent.

The other modules are the communicational trace and the mental model. The planning module allows the agent to create messages on the basis of the elements selected by the user. The task of the learning module is to learn new knowledge such as user's preferences or formulations used in a given organization. As outlined earlier, the communication module allows the agent to reason about the communication activity. At the beginning, the communication module receives the request from the planning module. During the rest of the communication, the communication module transforms negotiation positionings into corresponding CG-KQML+ messages and vice-versa by calling specific patterns from the different knowledge models (an example is given in next section "communication manager"). The natural language text is generated using static formulation schemas. Also, during the communication, all the positionings are recorded in the communicational trace.

Figure 4 gives a snapshot of the POSTAGE environment. A message is delivered from a user to another user using their respective POSTAGE agents. The user specifies his message by selecting the receiver and the subject of the message. The user clicks on the "Generate" button to ask the agent to generate a formulation which appears in the "Send" pane. Then, the user clicks on the "Send" button. When a user receives a message, in the "Reception" pane of his/her POSTAGE agent, s/he selects one of the possible positionings, "Accept" or "Refuse". During the correspondence, "help" messages are printed in the pane located in the middle of the window.

**The planner**

To each performative type corresponds a plan with preconditions that mirror the preparatory conditions[5] specified in speech act theory. For instance, when performing a directive, the locutor should ensure that his interlocutor has the capability to perform the requested action. For POSTAGE, if a user selects the subject "Subject: Prepare the syllabus" and the receiver "Receiver: Student Hamza", the POSTAGE agent will not generate the corresponding message and will print to the user "Hamza is not responsible for preparing syllabuses, please choose another subject or another receiver".

Thus, when the subject and the receiver are selected and sent to the planner, this latter identifies the performative act from the subject, looks for the plan associated to the identified performative act, checks the plan's preconditions and then executes the plan's actions. This treatment is naturally implemented using Prolog+CG :

```
ExecutePlan(_subject, _receiver) :-
       FindPerformative(_subject, _performative),
       [PLAN : _IdPlan]-
              -hasPerformative->[PERFORMATIVE = _performative],
              -hasDescr->[Description = _descrp],
              -hasCond->[Condition = _cond_list],
              -hasBody->[Body = _actions_list],
              -hasEffects->[Effects = _effects_list],
       VerifyPreconditions(_cond_list, _subject, _receiver),
       ExecuteActions(_actions_list, _subject, _receiver).
```

**The Communication manager**

The communication manager is called by the planner and it represents the user's communicative intention by creating the corresponding communicational state (CS) and the appropriate negotiation positioning.

Let's illustrates this treatment with the following exchange between a secretary (whose name is Nadia) and a professor (whose name is Adam).

```
Message from the secretary to the professor
       Mr. Adam, please come and take the copies of your exam.
Message from the professor to the secretary
       OK I am coming.
```

Using the POSTAGE environment, the secretary chooses the following items: "Receiver: Professor Adam", "Subject: Availability of the exam copies". Then the secretary clicks on the "Generate" button. The selected information are then sent to the planner of her agent.

The communication manager checks the power and social relation existing between the sender and the receiver and finds that they share a peer social power (from the user's model). To express the above communication act, the following conceptual structure is then created :

```
[POSIT : p1]-
 -hasType->[TYPE : propose],
 -hasSender->[AGENT : nadia],
 -hasReceiver->[AGENT : adam],
```

[5] Preparatory conditions determine which propositions a speaker would presuppose if he was performing a speech act [Vanderveken 90].

```
-timeOf->[Time : t1],
-hasPower->[SOCIAL_POWER : peer],
-hasCS->[CONTENT:
      [CS:cs1]-
        -hasType->[TYPE : directive],
        -hasSender->[AGENT : nadia],
        -hasReceiver->[AGENT : adam],
        -timeOf->[Time : t1],
        -hasContent->[CONTENT:
          [AGENT:adam]<-agnt-[COME]-purp->[ACTION:
              [AGENT:adam]<-agnt-[TAKE]-obj->[OBJECT:"exam copies"]],],].
```

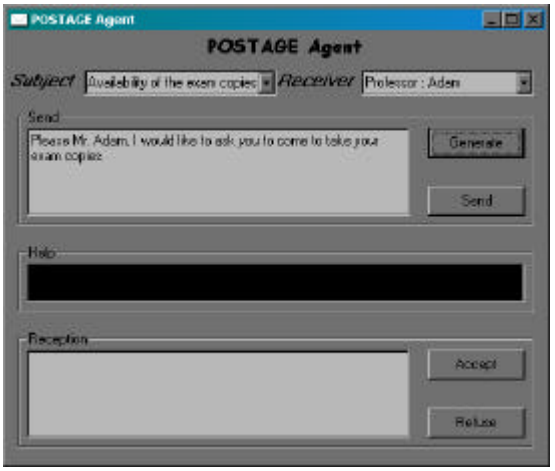


**Fig. 4**. Snapshop of the POSTAGE environment.

The agent of Nadia adds this CS in its communicational trace, finds a corresponding text formulation[7] which agrees: (1) with the social relationship and power between his user and the receiving user and (2) with the formulation rules used in an university. It then constructs the corresponding CG-KQML+ message which will be sent to the receiving agent:

```
[KQML+_MESSAGE]-
    -hasSender->[AGENT : nadia],
    -hasReceiver->[AGENT : adam],
    -hasPrimPerf->[DIRECT],
    -hasExprPerf->[ASK],
    -hasPresentation->[pleasure],
    -hasContent->[CONTENT:
       [AGENT:adam]<-agnt-[COME]-obj->[ACTION :
              [AGENT:adam]<-agnt-[TAKE]-obj->[OBJECT:"examcopies"] ],],
    -replyWith->[NUMBER : r1].
```

When the receiving agent receives the message, he first prints the text on the screen to be read by his user. The user has the choice of accepting or refusing the directive by pressing

[7] Currently, the communication module generates the message by using static formulation schemas from the static knowledge model. However, since the propositional content of CG-KQML+ messages are represented using conceptual graphs and natural language processing is a central topic in the CG community, we plan to reconsider the natural language side of our work in the future.

the corresponding button. In the present example, the user clicks on the "Accept" button. Then the agent A2 constructs the corresponding positioning of acceptance:

```
[POSIT : p2]-
 -hasType->[TYPE : accept],
 -hasSender->[AGENT : adam],
 -hasReceiver->[AGENT : nadia],
 -timeOf->[Time : t2],
 -hasPower->[SOCIAL_POWER : peer],
 -hasCS->[CONTENT :
                [CS:cs1]-
                   -hasType->[TYPE : directive],
                   -hasSender->[AGENT : nadia],
                   -hasReceiver->[AGENT : adam],
                   -timeOf->[Time : t1],
                   -hasContent->[CONTENT:
    [AGENT:adam]<-agnt-[COME]-obj->[ACTION:
     [AGENT:adam]<-agnt-[TAKE]-obj->[OBJECT:"exam copies"]],],].
```

The agent A2 constructs both the corresponding text of acceptance and displays it for his user. When the user clicks on the "Send" button, the agent also constructs the CG-KQML+ message to be sent to agent A1:

```
[KQML+_MESSAGE]-
    -hasSender->[AGENT : adam],
    -hasReceiver->[AGENT : nadia],
    -hasPrimPerf->[ACCEPT],
    -hasContent->[CONTENT:
       [AGENT:adam]<-agnt-[COME]-obj->[ACTION :
              [AGENT:adam]<-agnt-[TAKE]-obj->[OBJECT:"examcopies"]], ],
    -inReplyTo->[NUMBER : r1].
```

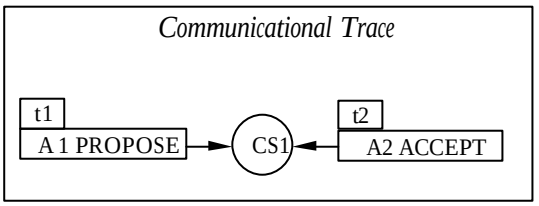


**Fig. 5.** Communicational trace of agent A1

Finally, when the agent A1 receives this CG-KQML+ message, he finds the corresponding text to be read by the user and translates it to a corresponding positioning. Figures 5 shows the communicational trace of agent A1.

## 5 Future works and conclusion

We showed some limitations of the KQML language regarding the expression of the various kinds of speech acts found in human conversations. Using more carefully speech act theory, we proposed an extension to KQML. CG-KQML+ is a new agent communication language that allows agents to express any kind of speech act and that takes into account social relationships. Furthermore and unlike KQML, the primitive CG-KQML+ performative verbs eliminate the possibility of having different interpretations for each performative type.

We illustrated the use of CG-KQML+ in the context of the multi-agent system POSTAGE. CG-KQML+ as well as the new version of POSTAGE makes an extensive use of the conceptual graph formalism, to represent various kinds of conceptual structures used in our system. Since the language Prolog+CG is a CG-based logic programming language, we used it to implement the new version of POSTAGE. Due to the abstract level provided by Prolog+CG, the second version of POSTAGE is more concise, readable and took much lesser time and effort than the first version. Future works include the use of POSTAGE in other domains, a deep treatment of the natural language processing side of our system, a deep treatment of agent' social and emotional features and finally, a development of a multi-agent system shell based on top of Prolog+CG.

## References


[1] M. Barbuceanu, M. S. Fox, The Specification of COOL: A Language for Representation Cooperation Knowledge in Multi Agent Systems, EIL, Univ. of Toronto, Internal Report, 1996.

[2] K. Bouzouba, Modélisation des interactions basée sur le point de vue des agents, Ph.D. Thesis, Laval University, Canada, 1998.

[3] K. Bouzouba, B. Moulin, KQML+ : An Extension of KQML in order to deal with Implicit Information and Social Relationships, In Proc. of FLAIRS'98, May, pp. 289-293, 1998.

[4] M. Busuioc, Distributed Intelligent Agents - A Solution for the Management of Complex Services, The Intelligent Agents for Telecom Applications (ECAI'96) Workshop Proc., 1996.

[5] A. Chavez, P. Maes, Kasbah: An Agent Marketplace for Buying and Selling Goods, Proc. of the First Intern. Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, April, 1996.

[6] J. Chu-Carberry, S. Carberry, Conflict Detection and Resolution in Collaboration Planning, Intelligent Agents II, LNAI 1037, Heidelberg: Springer-Verlag, 1995.

[7] P.R. Cohen, H.J. Levesque, Persistence, intention and commitment, In Cohen P.R., Morgan J., Pollack M.E. (Eds), Intentions in communication, MIT Press, 33-69, 1990.

[8] P. R. Cohen, H. .J. Levesque, Communicative actions for artificial agents, In Proc. of ICMAS'95, pp. 65-72, 1995.

[9] http://www.inprise.com/.

[10] F. Dignum, B. Van Linder, Modelling Social Agents: Communication as Action, In Intelligent Agents III : Agent, Theories, Architectures, and Languages, Lecture Notes in Artificial Intelligence 1193, J. P. Müller, M. J. Wooldridge, N. R. Jennings (Eds), Springer Verlag, 1997.

[11] http://www.haley.com/.

[12] T. Finin, Y. Labrou, J. Malfield, KQML as an agent communicaton language, Technical Report, Computer Science and Electrical Engineering, University of Maryland Baltimore County, Baltimore MD USA, September, 1995.

[13] http://drogo.cselt.stet.it/fipa

[14] A. Haddadi, *Communication and Cooperation in Agent Systems, A Pragmatic Theory*, Springer-Verlag, 1995.

[15] A. Kabbaj and M. Janta, From PROLOG++ to PROLOG+CG : an object-oriented logic programming, in Proc. of the Intern. Conf. ICCS'2000, Darmstadt, Germany, August, 2000.

[16] A. Kabbaj, B. Moulin, O. Rouleau, D. Nadeau and J. Gancet, Uses, Improvements and Extensions of Prolog+CG : Case Studies, Submitted to ICCS'2001.

[17] D. L. Martin, A. J. Cheyer, and D. B. Moran, The open agent architecture: A framework for building distributed software systems, *Applied Artificial Intelligence*, vol. 13, pp. 91-128, January-March 1999.

[18] B. Moulin, D. Rousseau, G. Lapalme, A Multi-agent approach for modelling conversations, In Proc. of the Intern. Conf. on Artificial Intelligence and Natural Language, Paris, June, 1994.

[19] B. Moulin, B. Chaib-draa, Distributed Artificial Intelligence: an overview, in N. Jennings, G. O'Hare editors, Foundations of Distributed Artificial Intelligence, Wiley 1996, 3-55.

[20] M. H. Nodine, A. Unruh, Facilitating Open Communication in Agent Systems: the InfoSleuth Infrastructure, The Fourth International Workshop on Agent Theories, Architectures, and Languages (ATAL IV), Providence, Rhode Island, July, 1997.
[21] H. Nwana, D. Ndumu, L. Lee, J. Collis, ZEUS: A Tool-Kit for Building Distributed Multi-Agent Systems, In Applied Artifical Intelligence Journal, vol 13, num 1, p129-186, 1999.
[22] J.R. Searle, Speech Acts, Cambridge University Press, 1969.
[23] N. Skarmeas, K. Clark, Process Oriented Programming for Agent-Based Network Management, The Intelligent Agents for Telecom Applications (ECAI'96) Workshop Proceeding, 1996.
[24] M.D. Sadek, Dialogues Acts are Rational Plans, Research Report, Centre National d'Etudes des Télécommunications, Lannion Cedex, 1990.
[25] Y. Shoham, Agent Oriented Programming, Artificial Intelligence, vol. 60 (1), pp. 51-92, 1993.
[26] J. F. Sowa, *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks Cole Publishing Co., Pacific Grove, CA, August, 1999.
[27] K. Sycara, D. Zeng, Coordination of Multiple Intelligent Software Agents, International Journal of Cooperative Information Systems, vol 5, num 2 & 3, 1996.
[28] D. Vanderveken, *Meaning and speech acts (vol.1), Principles of language use*, Cambridge University Press, 1990.