Extending the FIPA ACL Language. From Object Based Descriptions to Relational Representations

Luís Botelho[1] and Pedro Ramos[1]

[1]Department of Information Sciences and Technologies of ISCTE
Av das Forças Armadas, 1600 Lisboa, Portugal
{Luis.Botelho, Pedro.Ramos}@iscte.pt

Abstract. This paper describes a formalism to represent object-descriptions within inter-agent communication messages and a formalism to express the relational representation and the data model of those descriptions. A mechanism for generating the data model and the relational representation of an object from its object-based description is also proposed. This mechanism is used to support the enhancement of FIPA ACL inter-agent communication language with three new communicative acts: *present-object*, *ask-object* and *subscribe-object*. The paper presents the intentional semantics of *present-object* and *ask-object* messages.

1 Introduction

KQML [6] is the *de facto* standard in inter-agent communication. However, since KQML has been criticized due to its poor semantics, FIPA (Foundation for Intelligent Physical Agents) has recently proposed ACL, a speech act based language [8] as a new standard for inter-agent communication. ACL [5] includes, among others, some informative message types (*inform*, *confirm*, *disconfirm*) for which the content must be a proposition. However, for some applications (mainly, applications based on the representation of multimedia objects, such as video-based surveillance applications [4]), the most natural way of conveying information is to use objects, not propositions. Indeed, in monitoring applications (Figure 1) some agents extract object-descriptions from observed physical processes and send them to other agents. Such application domains would benefit from the development of formal methods for communication, representation and reasoning based on objects rather than on propositions.

ACL is a recent language whose continuous evolution has been documented in several versions of the FIPA specifications [4][5]. Here, we present yet another evolution.

We propose to extend the FIPA ACL language with three new messages that may be used by agents to send and to ask for descriptions of objects: *present-object*, *ask-object*, and *subscribe-object*. *present-object* is an informative speech act that is

used to present an object to an agent. *present-object* takes an object as content whereas the other ACL informative messages take propositions.

The sender may reasonably expect that the presentation of an object to the receiver will change its mental state. Upon being presented object *obj* by means of a *present-object* message, an agent is expected to acquire the new belief that *obj* is an object. It is also expected that the receiver forms some other beliefs about the class and the attributes of the object.

This paper presents a formal approach to deal with the representation of object descriptions and an automatic mechanism that takes an object description and computes a set of first order predicate calculus sentences that represents the object and its correspondent data model. Using this approach it is possible to define the intentional meaning of the new proposed messages.

The remaining of the paper is organized as follows. The second section motivates the work through an example. In the third section, we discuss the formal representation of object descriptions. The fourth section describes the formal mechanism used to produce the data model and the relational representation of an object from the object description. In the fifth section we present the new messages and their intentional semantics. The sixth section presents conclusions and future work.

2 Motivation

MODEST [2][1] is a European ACTS project whose development goal is to define and build a video-based intelligent multi-agent system for traffic surveillance (Monitorix). As previously pointed out in [2], a great deal of inter-agent communication in Monitorix follows the communication pattern described in Figure 1.

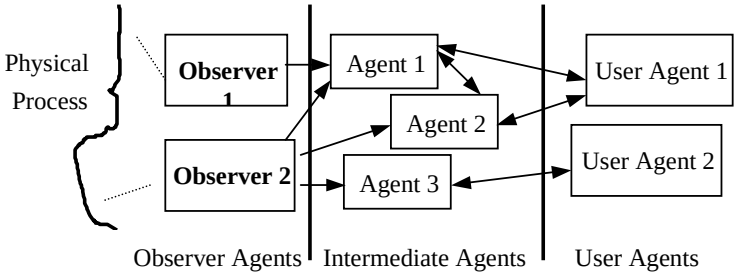

Figure 1 - Reference Model for Monitoring Multi-Agent System

In the class of MAS described in Figure 1, there is a physical process that is observed, in real-time, by a set of agents (observer agents). These agents extract objective object descriptions from the physical process and deliver them to the intermediate agents that build a higher-level relational representation of the observed process. Following an adequate protocol, an information requester (e.g., an intermediate agent) subscribes

certain information classes with one or more information providers (e.g., observer agents). Generally, this subscription is done only once during the initial stage of the agents' life. The providers repeatedly send instances of the subscribed information classes to the requesters. Usually (although not always), the information sent by the providers to the requesters consists of object descriptions. Figure 2 illustrates the kind of information message sent by one of the observer agents to an intermediate agent. SL (Semantic Language) [5] is the content language used in the examples given in the paper.

(a)

```
(query-ref
 :sender IntermediateAgent-1
 :receiver ObserverAgent-1
 :content
  (any ?x (mobile-object ?x))
 :language SL)
```

(b)

```
(inform
 :sender  ObserverAgent-1
 :receiver IntermediateAgent-1
 :content   (=
  (any ?x (mobile-object ?x))
  (Vehicle
   :vehicle_color red
   :vehicle_position
    (Position :y 1253 :x 34)
   :vehicle_size
    (Size :length 4 :width 2)))
 :language SL)
```

Figure 2 – Agent interaction in a monitoring multi-agent system

The message depicted in Figure 2(b) has the following reading *"one of the entities that satisfy the condition of being a mobile object is a red vehicle..."*.

We would like that, after having received the message in Figure 2(b), any intermediate agent would be able to answer questions such as

1) What is the class of the received mobile-object? ; 2) What are the attributes of a vehicle? 3) What is the position of the red vehicle?

Figure 3 depicts a message used by an agent to ask the position of a red vehicle.

```
(query-ref :sender i :receiver j
   :content
      (any ?pos (exists ?size
                  (mobile-object(Vehicle
                     :vehicle_color red
                     :vehicle_position ?pos
                     :vehicle_size ?size)))))
```

Figure 3 – What is the position of the red vehicle?

Although it is possible to ask the above question, it would not be practical to do it if the class of vehicles had many attributes, each one being a composed object itself (as actually it is the case in the Monitorix system). We would have to use an existential

variable as a place holder for the value of each irrelevant attribute (such as *?size* in Figure 3). While it is feasible (although cumbersome) to ask questions such as question number 3 above, it is impossible to ask questions number 1 and 2. The class of the object and the names of its attributes are not explicitly available. We would have to use domain dependent knowledge to access the class and the names of the attributes of an object. Instead of using domain-dependent knowledge, we propose a general-purpose approach. Using the data model described in Figure 4, Intermediate Agent 1 would be able to answer meta-questions about vehicles in general. Through the first and second queries of Figure 6, User Agent 1 asks Intermediate Agent 1 *"what are the known classes?"* and *"what are the attributes of a vehicle?"*.

```
(class vehicle)
(class position)
(class size)
(attribute vehicle
   vehicle_color)
(attribute vehicle
   vehicle_position)
(attribute vehicle
      vehicle_size)
(attribute position position_x)
(attribute position position_y)
(attribute size size_length)
(attribute size size_width)
(class-relation vehicle
   vehicle_position position one)
(class-relation vehicle
      vehicle_size size one)
```

Figure 4 - Data Model of the Class Vehicle

Abbreviations

```
v: (Vehicle
   :vehicle_color red
   :vehicle_position
      (Position
         :position_y 1253
         :position_x 34)
   :vehicle_size
      (size
         :size_length 4
         :size_width 2))
```

```
p:(Position
   :position_y 1253
        :position_x 34)
```

```
s: (Size
   :size_length 4
   :size_width 2)
```

Representations

```
(instance vehicle v)
(instance position p)
(instance size s)
(value v vehicle_color red)
(value v vehicle_position p)
(value v vehicle_size s)
(value p position_x 1253)
(value p position_y 34)
(value s size_length 4)
(value s size_width 2)
```

Figure 5 - Relational representation of a vehicle

The model is described using the relations `class/1`, `attribute/2` and `class-relation/4`. The relation `attribute` lists the attributes of a class. The relation `class-relation` is used when the value domain of a class attribute is another class. The cardinality of the relation between these two classes (e.g., Size and Vehicle) is also represented. For example, `(class-relation vehicle vehicle_size size one)` means that one instance of a vehicle can be related

(through the attribute `vehicle_size`) with no more than one instance of size. Notice that the *reverse side* of the cardinality can only be determined through the analysis of a set of vehicles (if we observe two vehicles with the same size, the following relation would be inferred: `(class-relation size vehicle_size vehicle many)`). This kind of inferences is not treated in this paper. In section 4 we present an inference mechanism that automatically produces the descriptions presented in Figure 4 and Figure 5.

Given the relational description in Figure 5, an user agent may ask Intermediate Agent 1 *"what is the position of a red vehicle?"*, through the third query of Figure 6. The relational description is expressed with the relations *instance/2* and *value/3*. In the example of Figure 5 `(instance size s)` means that s is an instance of the class `size` and `(value s length 4)` means that 4 is the value of the attribute `length` in the instance `s`.

// What classes are there?

```
(query-ref
  :sender UserAgent-1
  :receiver IntermediateAgent-1
  :content
   (all ?x (class ?x))
  :language SL)
```

// What are the attributes of vehicle?

```
(query-ref
  :sender UserAgent-1
  :receiver IntermediateAgent-1
  :content
   (all ?x
     (attribute vehicle ?x))
  :language SL)
```

// What is the position of a red vehicle?

```
(query-ref
 :sender UserAgent-1
 :receiver IntermediateAgent-1
 :content
  (any ?x (exists ?v
   (and
    (instance vehicle ?v)
    (value ?v vehicle_color red)
    (value ?v vehicle_position ?x)
   )
  )
 :language SL)
```

Figure 6 - Query messages in the Monitorix system

The work reported in this paper has two motivations. The first is to define an automatic mechanism that takes a description such as the one in the message of Figure 2(b) and produces a set of first order logic sentences representing the data model of an agent and its relational representation (such as in Figure 4 and Figure 5). These sentences enable the receiver to answer questions such as in Figure 6. The proposed mechanism is not tied to our particular application domain. Other domains in which the fundamental pieces of information are objects will equally profit from this automatism. Second, we propose to extend the FIPA ACL language with new messages for handling descriptions. The expected effects and the preconditions of these messages are easily described using the formalism and computation mechanism presented in the paper.

3 Representing Objects

In monitoring and in video-based applications, objects are the fundamental data entity. Assuming a logic-based approach to representation and communication, we view objects as compound terms. An object can be represented as a functional expression in which the function symbol is the constructor of the object class and the arguments are the values of the attributes of that object. Instances of the classes *vehicle* and *shooting conditions* would be represented by functional expressions. The constructor of the class *vehicle* could be the function *Vehicle*, and the constructor of the class *shooting conditions* could be the function *ShootingCondition*. *Vehicle* takes the three arguments *vehicle_color*, *vehicle_position* and *vehicle_size* in this order. *ShootingConditions* takes four arguments: *luminance*, *tilt*, *time* and *day* in this order.

Description (d1)

```
(Vehicle
   :vehicle_color red
   :vehicle_position
      (Position
         :position_y 1253
         :position_x 34)
   :vehicle_size
      (Size
         :size_length 4))

(Vehicle red
    (Position 1253 34)
    (Size 4 unspecified-width))
```

Description (d2)

```
(ShootingConditions
   :tilt 0
   :luminance 125
   :time 2000/06/15
   :day 8:17:13)

(ShootingConditions
   0
   125
   2000/06/15
   8:17:13)
```

Figure 7 - Description of a vehicle and a shooting condition

In many applications in which the data collected from the sensors is not accurate, the confidence associated with each individual piece of information can be very low. Sometimes, the confidence of a specific component of an object is so low that the agent that requests that class of objects does not want to receive that component (this allows us to avoid flooding the communication channels with useless data). For this reason, we use a special notation for the functional expressions that represent objects. This notation consists of explicitly stating the names of the attributes of each object. For instance, the color of a vehicle will be preceded by the special identifier *:vehicle_color*, the position will be preceded by the identifier *:vehicle_position* and the size will be preceded by *:vehicle_size*. Functional expressions representing objects, written in the special notation just described, are called descriptions. Figure 7 shows two examples of descriptions and the equivalent functional expressions written in the usual notation. In Figure 7, (d1) is a partial description of a vehicle in which the width of the size of the vehicle is not specified. (d2) is the description of the shooting conditions in a particular instant of time. The main advantage of using this notation is that it allows the representation and communication of incomplete descriptions. A minor advantage is

that the order of the components of a description is not predetermined thus allowing greater flexibility in inter-agent communication.

4 A Calculus for Object Descriptions

In this section we propose a set of inference rules that can be used to compute the relational representation and the data model of an object from its description.

We introduce the special predicate *description/1* that is true when its argument is the description of an object that belongs to the domain of the discourse. As described in section 3, objects are represented by functional expressions in which the constructor of the class of the object is applied to the values of the attributes of that object: (*constructor component$_1$ … component$_n$*).

Let D be the set of all objects of the domain of the discourse, R be the set of all relations and F be the set of all functions. Let also c represent the constructor of a class of objects, φc represent the name of the class whose constructor is c, and v_i represent the value of the attribute a_i. The interpretation function I maps each constant symbol into an element of D, each relation symbol into an element of R and each function symbol into an element of F. Iσ represents the application of I to σ. The formula *(description (c :a$_1$ v$_1$ … :a$_n$ v$_n$))* is satisfied iff $Ic(Iv_1, \ldots, Iv_n) \in D$.

The mechanism used to build the relational representation and the data model of an object from its description is supported by the following set of inference rules:

Data-Model Rules

If (c :a_1 v_1 :a_2 v_2 … :a_n v_n) is a description of an object of class φc, then φc is a class

(description (c :a_1 v_1 :a_2 v_2 … :a_n v_n)) (Rule 1)

(class φc)

If (c :a_1 v_1 :a_2 v_2 … :a_n v_n) is a description of an object of class φc, then a_i is an attribute of class φc.

(description (c :a_1 v_1 :a_2 v_2 … :a_n v_n)) (Rule 2)

(attribute φc a_i) for any i=1, …, n

Let Ωv denote the class of v and Ψv be a functional expression that returns the constant *many* when v is a list and returns the constant *one* otherwise.

If (c :a_1 v_1 :a_2 v_2 … :a_n v_n) is a description of an object of class φc, then the attribute a_i defines a one to Ψv_i relation (i.e., a one to one or a one to many relation) between φc and the class Ωv_i.

(description (c :a_1 v_1 :a_2 v_2 … :a_n v_n)) (Rule 3)

(class-relation φc a_i Ωv_i Ψv_i) for any i=1, …, n

Relational Representation Rules

If (c :a_1 v_1 :a_2 v_2 … :a_n v_n) is a description of an object of class φc, then (c :a_1 v_1 :a_2 v_2 … :a_n v_n) describes an instance of the class φc.

(description (c :a_1 v_1 :a_2 v_2 … :a_n v_n)) (Rule 4)

(instance φc (c :a_1 v_1 :a_2 v_2 … :a_n v_n))

If (c :a_1 v_1 :a_2 v_2 … :a_n v_n) is a description of an object of class φc, then v_i is the value of attribute a_i of (c :a_1 v_1 :a_2 v_2 … :a_n v_n).

(description (c :a_1 v_1 :a_2 v_2 … :a_n v_n)) (Rule 5)

(value (c :a_1 v_1 :a_2 v_2 … :a_n v_n) a_i v_i) for any i=1, …, n

Recursive Rule

If the value v of an attribute is a description, then (description v) is true.

(description (c :a_1 v_1 :a_2 v_2 … :a_n v_n)) (Rule 6)

(description v_j) for any j=1, …, n such that v_j is a description

Notice that this last rule allows the recursive generation of the relational representation and the data model of the descriptions that appear as values of the attributes of objects.

If the previous rules are applied to the description contained in the message depicted in Figure 2 we obtain the relational representation shown in Figure 5 and the data-model appearing in Figure 4.

5 An Extension to FIPA ACL Language

Given the formalism described in sections 3 and 4, it is now possible to extend the FIPA ACL inter-agent communication language with three new messages related to the object-based communication.

We propose to adopt the message *present-object* for an agent to present the description of an object to another agent. *present-object* has the following basic syntax:

```
(present-object :sender I :receiver j :content <Description>)
```

in which <Description> is a description of an object as was discussed in section 3.

The intentional semantics of a message consists of defining the condition that has to hold just before the agent sends it (feasibility precondition) and the state that the sender may reasonably expect to hold after the message has been received (rational effect).

Both the feasibility precondition and the rational effect are specified in terms of the agent's mental states [7]. Agent's mental states are similar to those of BDI architectures [3]. They are described using several modal operators such as *bel*, *bel-if*, *uncertain-if* and *intends* [5]. Informally, (bel α ϕ) means agent α believes ϕ, (bel-if α ϕ) means agent α believes ϕ or it believes $\neg\phi$, (uncertain-if α ϕ) means agent α is uncertain about ϕ or it is uncertain about $\neg\phi$, and (intends α ϕ) means agent α intends to be in a state in which ϕ holds.

Feasibility precondition of *present-object*. An agent can only send the *present-object* message <*sender*, present-object(*receiver*, δ)> if it believes that δ describes an existing object and it does not believe the receiver knows anything about the described object:

```
(and (bel sender (description δ))
   (not (bel sender (or (bel-if receiver (description δ))
                        (uncertain-if receiver (description δ))))))
```

Rational effect of *present-object*. An agent selects a message to send if it may reasonably expect the message to produce some changes in the receiver's mental state. The sender of the *present-object* message <*sender*, present-object(*receiver*, δ)> may reasonably except that the receiver will come to believe δ describes an existing object:

```
(bel receiver (description δ))
```

(a)

```
(inform
 :sender ObserverAgent-1
 :receiver IntermediateAgent-1
 :content
   (=
    (any ?x (mobile-object ?x))
    (Vehicle
      :vehicle_color red
      :vehicle_position
        (Position
          :position_y 1253
          :position_x 34)
      :vehicle_size
        (Size
          :size_length 4
          :size_width 2)))
 :language SL)
```

(b)

```
(present-object
  :sender ObserverAgent-1
  :receiver IntermediateAgent-1
  :content
    (vehicle
      :vehicle_color red
      :vehicle_position
        (Position
          :position_y 1253
          :position_x 34)
      :vehicle_size
        (size
          :size_length 4
          :size_width 2)))
  :language SL)
```

Figure 8 - Comparison between Inform and present-object message

Figure 8 shows how the *inform* message presented in Figure 2(b) can be replaced by a more natural one using the *present-object* message. After receiving the message of Figure 8(b) Intermediate Agent 1 will be able to answer all the questions mentioned in section 2 using the inference rules proposed in section 4. The application of the rules to

the description presented in Figure 8(b) is described in Figure 9. We are assuming the receiver creates the belief (description (Vehicle …)) as specified by the *present-object* message.

1	(description v)	Effect of Msg.	13	(attribute position position_x	R2, 10
2	(class vehicle)	R1, 1	14	(attribute position position_y)	R2, 10
3	(attribute vehicle vehicle_color)	R2, 1	15	(instance position p)	R4, 10
4	(attribute vehicle vehicle_position)	R2, 1	16	(value p position_x 1253)	R5, 10
5	(attribute vehicle vehicle_size)	R2, 1	17	(value p position_y 34)	R5, 10
6	(instance vehicle v)	R4, 1	18	(class-relation vehicle vehicle_position position one)	R3, 10
7	(value v vehicle_color red)	R5, 1	19	(class size)	R2,11 1
8	(value v vehicle_position p)	R5, 1	20	(attribute size size_length)	R2, 11
9	(value v vehicle_size s)	R5, 1	21	(attribute size size_width)	R2, 11
10	(description p)	R6, 1	22	(instance size s)	R4, 11
11	(description s)	R6, 1	23	(value s size_length 4)	R5, 11
12	(class position)	R1, 10	24	(value s size_width 2)	R5, 11
			25	(class-relation vehicle vehicle_size size one)	R3, 11

v, p and s are the same abbreviations used in Figure 5

Figure 9 - Using Inference Rules to build a Data Model and a Relational Representation

We also propose *ask-object* as the dual message of *present-object*. *ask-object* is used by an agent to ask another agent for the description of an object that matches the condition specified by an identifying expression. In the message of Figure 10, User Agent 1 asks Intermediate Agent 1 to present it the descriptions of all the vehicles with length greater than 4 meters. As a result, Intermediate Agent 1 will use a *present-object* message.

```
(ask-object                    (ask-object
 :sender i                      :sender UserAgent-1
 :receiver j                    :receiver IntermediateAgent-1
 :content                       :content
  <identifying expression>)      (all ?obj (exists ?x (exists ?size
                                    (and (instance ?obj vehicle)
                                      (value ?obj vehicle_size ?size)
                                      (value ?size size_length ?x)
                                      (> ?x 4))))))
```

Figure 10 - Example of an ask-object message

Finally we propose the *subscribe-object* message as the persistent version of the *ask-object* message. After receiving a *subscribe-object* message the receiver is expected to form the intention of presenting the specified objects to the requester whenever there are new objects available that satisfy the specified condition. Although the syntax of *ask-object* and *subscribe-object* is exactly the same as the syntax of *query-ref* and *subscribe*, the rational effect of *ask-object* is different from the rational effect of *query-ref* and analogously for *subscribe-object* and *subscribe*. The rational effect of *query-ref* is to create the intention of sending an *inform* message, whereas the rational effect of *ask-object* is to create the intention of sending a *present-object* message. In the remaining of this section, we present the feasibility precondition and the rational effect of the *ask-object* message. The *subscribe-object* is the persistent version of *ask-object*.

Feasibility precondition of *ask-object*. An agent can only send the message <*sender*, ask-object(*receiver*, *expr*)> if it doesn't know anything about the specified object. Furthermore, the sender must believe the receiver does not have the intention to present it the specified object yet:

```
(and  (not (bel-ref sender expr))
      (not (uncertain-ref sender expr))
      (not (bel sender
             (intends receiver (done
              (present-object
                 :sender receiver :receiver sender :content expr))))))
```

Rational effect of *ask-object*. An agent sends the message <*sender*, ask-object(*receiver*, *expr*)> if it wants the receiver to send it a *present-object* message with the object satisfying the specified condition (in which | is the alternative action operator and obj_i is the description of an object known by the receiver agent.):

```
(done (|
      (present-object :sender receiver :receiver sender :content obj1)
            …
      (present-object :sender receiver :receiver sender :content objn)))
```

6 Conclusions and Future Work

The main contribution of the paper is a formalism for expressing the relational representation and the data model of objects, and a rigorous inference mechanism for generating them from object descriptions. The other main contribution is the proposal to extend FIPA ACL language with three new communicative acts. These messages are especially important in environments in which the processing time and/or the bandwidth are at premium because they allow sending and processing simpler and

shorter messages. Besides, they allow improving communication when the natural entities of the domain are objects, not propositions.

The proposals presented in this paper contribute to get people with object-oriented background (such as the MPEG7 community) closer to people with logic background.

In future work, we intend to extend the semantics of the predicate calculus in order to accommodate the special predicates *class*, *instance*, *class-relation*, *value* and *attribute*. The formal semantics of the *subscribe-object* message must also be defined. The inference mechanism has to be completed in order to derive all instances of the *class-relation* special predicate.

Acknowledgement. We want to acknowledge the Modest Project for providing concrete examples of an intelligent multi-agent system and of inter-agent messages.

References

1. Abreu, B.; Botelho, L.M.; Cavallaro, A.; Douxchamps, D.; Ebrahimi, T.; Figueiredo, P.; Macq, B.; Mory, B.; Nunes, L.; Orri, J.; Trigueiros, M.J.; and Violante, A.: Video-Based Multi-Agent Traffic Surveillance System. Proceedings of the IEEE Intelligent Vehicle Symposium (2000)
2. Botelho, L.M.; Lopes, R.J.; Sequeira, M.M.; Almeida, P.F.; and Martins, S.: Inter-agent communication in a FIPA compliant intelligent distributed dynamic-information system. Proceedings of the International Conference on Information Systems Analysis and Synthesis (1999)
3. Cohen, P.R.; and Levesque, H.: Intention is choice with commitment. Artificial Intelligence. 42 (1990) 213-261
4. Foundation for Intelligent Physical Agents: FIPA 97 Specification, Part2 Agent Communication Language. (1997)
5. Foundation for Intelligent Physical Agents: FIPA 97 Specification, Version 2.0 Part2 Agent Communication Language. (1998)
6. Labrou, Y.; and Finin, T.: A semantics approach for KQML -- a general purpose communication language for software agents. Proceedings of the third International Conference on Information and Knowledge Management (1994)
7. Sadek, M.D.: Attitudes mentales et interaction rationnelle: vers une théorie formelle de la communication. Thèse de Doctorat Informatique, Université de Rennes I, France (1991)
8. Searle, J.R.: Speech Acts. Cambridge University Press (1969)