Operational Specification of a Commitment-Based Agent Communication Language*

Nicoletta Fornara
University of Lugano
via Buffi 13, 6900 Lugano, Switzerland
IDSIA
Galleria 2 6928 Manno, Switzerland
nicoletta.fornara@lu.unisi.ch

Marco Colombetti
University of Lugano
via Buffi 13, 6900 Lugano, Switzerland
Politecnico di Milano
Piazza L. da Vinci 32, I-20133 Milano, Italy
marco.colombetti@lu.unisi.ch

ABSTRACT

In this paper we propose an operational method to express the meaning of the messages exchanged among agents that interact in open environments. In an open environment, like for example the Internet, agents are usually designed by different constructors, so it is very important to define the meaning of a standard, widely accepted Agent Communication Language. We express the meaning of messages using the social notion of commitment. Commitments are defined operationally within an object-oriented paradigm. We give an operational specification of the commitment class that includes the concepts of conditional commitment and precommitment. Then we use commitment objects to define the meaning of some interesting speech acts, and give an example of their use in negotiation.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Languages and structures, Multiagent systems*

General Terms

Design, Standardization, Languages, Theory

Keywords

agent communication language, commitment, speech act, conversation protocol

1. INTRODUCTION

The possibility for different agents to interact in an open environment heavily depends on the adoption of a common, standard Agent Communication Language (ACL). The definition of a suitable ACL has therefore been widely recognized as a key step for the development of truly operative multiagent systems.

Like other existing proposal, our approach to the definition of a standard ACL is based on *speech act theory* [1], [8]. This approach to the study of communication views language use as a form of action, making it possible to treat communicative acts and other types of action in a uniform way. Speech act theory appears so suitable to describe communicative interactions among artificial agents that practically all existing proposals in the field of ACLs are based on it. Moreover, given that it provides an adequate approach to human communication, speech act theory allows one to treat communication among artificial and human agents in a uniform way - a crucial point to obtain successful mixed interactions among human beings and software agents.

Existing studies on ACLs follow three main approaches. The first approach, which we can call *mentalistic*, defines the meaning of a speech act using agent's *mental states*, like beliefs, desires and intentions. Examples of this approach are KQML [5] and FIPA ACL, proposed by the Foundation for Intelligent Physical Agent (FIPA) [6]. Using mental states to define speech acts may be adequate in *cooperative* multiagent system, but presents some problems when the multiagent system is composed by *competitive*, heterogeneous agents made by different vendors [10]. In this kind of context it is impossible to trust other agents completely or to make strong assumptions about their internal way of reasoning. It is therefore necessary to consider social, objective consequences, and new obligations of performing a speech act. In the *social* or *commitment-based* approach the meaning of a speech act is expressed using *commitments* directed from one agent to another. Formal proposals to treat speech acts in terms of commitments can be found in [3], [4], [12], [13]. Finally in the *conversational* approach the meaning of a speech act is implicitly defined as the role it plays in a given set of *conversation protocols* [7]. A problem with this approach, however, is that any change in the set of accepted protocols is going to affect the meaning of speech acts.

Our proposal is situated within the social approach, and is based on an analysis of the primitive notion of commitment. In particular we analyze the evolution of commitments through time, from precommitment or conditional

*Paper ID: 28, "STUDENT PAPER"

AAMAS'02, July 15-19, 2002, Bologna, Italy.

commitment to active commitment, then to fulfilled or violated commitment. We shall give an operational specification of commitment as an abstract data type. The commitment concept will then be used to define the meaning of a wide class of speech acts using a homogeneous framework. An important feature of our proposal is that it is modular and allows for the reuse of various components. Starting from a small set of basic operations on commitments, is possible to define the meaning of simple speech acts, which in turn can be used to define a new layer of more complex speech acts, and so on. In this way, ACL messages are given a formal semantics, thus eliminating any ambiguity in their use. Moreover, it becomes possible to check the soundness of conversation protocols defined in terms of speech acts. This is particularly important when the performance of communicative acts may have legal consequences, like for example in electronic auctions.

The paper is structured as follows. In Section 2, we introduce the main concepts used throughout the paper. In Section 3 we give the operational specification of commitment. In Section 4 we define the main speech act types. In Section 5 we illustrate the use of commitments objects in a realistic negotiation protocol. Finally, in Section 6 we discuss some relevant aspects of our work.

2. MAIN CONCEPTS

As previously mentioned, we want to propose an operational method to express the *meaning* of messages of an Agent Communication Language, suitable for agents that interact in *open environments*.

Agents interacting in an open environment form a kind of *society*. These agents are usually self-interested and heterogeneous. Self-interested means that agents from different constructors have the goal of maximizing their own utility. Heterogeneous means that agents are designed and developed by different vendors, and thus have different internal structures. Like with human beings, in order for agents to effectively interact they need: (i) to trust that other agents will actually play their role in the social structure; and (ii) to rely on a normative system, according to which unacceptable behavior can be managed.

Our proposal to satisfy these requirements is to use the notion of *social commitment*. The function of such commitments is to "stabilize" the interactions among agents. This is possible because commitments create the expectation that the other agents will behave in certain ways, and a means to deal with situations in which such expectations are not met.

2.1 Social Commitment

In this paper, we take an operational approach to the definition of commitments. In particular, we treat commitments introducing an abstract data type: the *commitment class*. We shall define its characteristics, its structure, its dynamic evolution in time, and the methods available for its manipulation. An instance of this class will be called a *commitment object* (see Section 3.1).

A commitment object is independent of an agent's internal structure and mental states, an important property to manage interactions among heterogeneous agents. It is *observable* by other agents and by any superordinate entity and it is *objective*. In this way every agent can objectively establish whether an agreement has been violated. Collections of commitment objects are used to describe the current state of the environment and to keep trace of the previous history of the system.

Intuitively a social commitment object represents a commitment made by an agent (the *debtor*), relative to another agent (the *creditor*), that some fact holds or some action will be carried out (the *content*) [2].

In a commitment object, the debtor and the creditor are represented by two agent identifiers. The content is represented by a formal statement that may take different truth values and is relative to a certain time period, during which the fact is due to hold or the action is due to be carried out. We formalize this notion as a *temporal proposition* (see Section 2.4).

A social commitment has a dynamic evolution in time. We keep trace of this evolution using what we call the *state* of the commitment. A commitment is *active* when it has not yet been fulfilled or violated; this is the case when the truth value of the commitment's content is still undefined. An active commitment may become *fulfilled* (when its content becomes true) or *violated* (when its content becomes false). Finally a commitment may be *cancelled*, and this simply means that it does not exist any more.

The creation and manipulation of commitments are based on the following operations: *make*, that establishes a new active commitment; the agent that performs this operation becomes the debtor of the commitment; *cancel*, that sets the commitment's state to *cancelled*; this operation can only be performed by the creditor of an existing commitment. Other operations, like the transfer of a commitment to another debtor or another creditor, are possible, but will not be dealt with in this paper.

Further some event driven routines are used to automatically update the state of commitment objects. In particular: a commitment's state is set to *fulfilled* when its content becomes true, and is set to *violated* when its content becomes false.

2.2 Conditional commitment

The concept described so far, which we may call absolute commitment, is sufficient to account for basic situations. However in many applications, like for example in electronic commerce, agents need to make commitments not in absolute terms, but under given conditions. For example an agent may commit to paying a sum of money to another agent only after it has received a particular product. We shall call this *conditional commitment*.

Conditional commitments are similar to absolute commitments, but have a further temporal proposition as their *condition*. When a conditional commitment is created its state is set to *conditional*; then the commitment becomes active if the condition is satisfied and becomes empty if the condition is not satisfied.

2.3 Precommitment

Using commitment objects as defined so far, it is possible to express the meaning of various speech acts like *assertions* and *promises*; however, it is not possible to express the meaning of directive speech acts, like *requests* and *questions*. When an agent requests another agent to do something, it is trying to induce the other agent to make a commitment. In fact usually when the context of the interaction does not define a relationship of authority between the interacting

agents, it is impossible for an agent to create a commitment of which some other agent is the debtor; in other words, one cannot directly commit somebody else. To solve this problem it is important to notice that by making a request, an agent sets up a situation in which the hearer can make a commitment just by *accepting* the request. For example with the request "Can you bring this book to John?" the speaker tries to induce the hearer to commit to the future performance of an action. After receiving the request, the hearer can create such a commitment just by saying "Yes, sure". This example shows that a request creates a particular social situation between the speaker and the hearer, that we call a *precommitment*. Precommitments are treated as a new state of commitment objects. They are created by directive speech acts; moreover, they can be transformed into active commitments by an act of *acceptance*, or cancelled by an act of *rejection*.

Some precommitments create conditional commitments if they are accepted: consider for example the request: "Would you please close the window as soon as it rains?". We shall simply represent these as precommitments with a nonempty condition field.

2.4 Temporal proposition

To express the content or the condition of a commitment we need to introduce a new abstract data type: the *temporal proposition class* (see Section 3.2). It includes a statement that may be in one of three different states: true, false or undefined. Moreover, to be used as the content of a commitment, a statement is typically relative to a time period. This time period may be the entire life of the system (*always*) or a finite time interval, including the limit case of a single instant. Statements may relate to the associated time period in two different temporal modes: they may be due to be true for the whole time period (like for example in the assertion "the price of this good is going to be 5$ for one week from now"); or they may be due to become true at some moment within the time period (like for example in the assertions "it rained yesterday" and "it will rain tomorrow"). In this way we cover a wide spectrum of commonly used propositions.

3. TECHNICAL SPECIFICATION

Our open interaction system consists of:

- A variable group of registered agents $\{a, b, ...\}$; a procedure is provided by which a new agent can register and join the group.
- A variable set of commitment objects $\{C_1, C_2, ...\}$; each commitment object is an instance of the commitment class described below.
- A variable set of temporal proposition objects $\{P, Q, ...\}$; each temporal proposition object is an instance of the temporal proposition class described afterwards. Temporal proposition objects are used to express the content or the condition of commitment objects, and their dynamic evolution is used to check if a commitment is fulfilled or violated.
- A fixed set of commitment-related actions that the agents are able to perform and whose meaning can be expressed in term of operations on commitments objects and proposition objects.
- A fixed set of event driven routines that we call *update rules*, which automatically update the state of commitment objects as a function of the state of the commitment's content or condition.

The global state s_t of this open system at time t, is composed by the registered agents and by the proposition and commitment objects. When an agent performs an action, the system evolves to another global state s_{t+1}, where the internal states of commitment and proposition objects are changed according to the effects of the action performed and by the update rules.

3.1 The commitment class

In this section we give a formal description of the commitment abstract data type. Following the object oriented paradigm, we formalize it as a class, defining the private fields that characterize a commitment object and the public methods used to manipulate it.

3.1.1 Fields

Identifier is the univocal identifier of the object in the system.

Debtor is the name of the registered agent that makes the commitment.

Creditor is the name of the registered agent relative to which the commitment is made.

State represents the dynamic evolution of the commitment object. It can assume one of following values:

- empty (e)
- cancelled (c)
- precommitment (p)
- conditional (cc)
- active (a)
- fulfilled (f)
- violated (v)

Figure 1 shows the finite state machine that describes the allowed dynamic evolution of the state of a commitment object. Dotted lines represent state changes caused by update rules. Solid lines represent state changes caused by the invocation of a method of the commitment class.

Content is a temporal proposition object (see below), representing the fact or the action the debtor is committed to, together with its temporal qualification and current state.

Condition is a temporal proposition object (see below). It represents the condition that must be satisfied, within the interval of time indicated, in order that the commitment becomes active. Between the interval of time of the content field and of the condition field there is a temporal constraint: the time interval of the condition must precede the time interval of the content.

Time-out is used only when the commitment's state is precommitment. It expresses the duration of the life of the precommitment. After the time-out is passed, the precommitment is automatically cancelled.

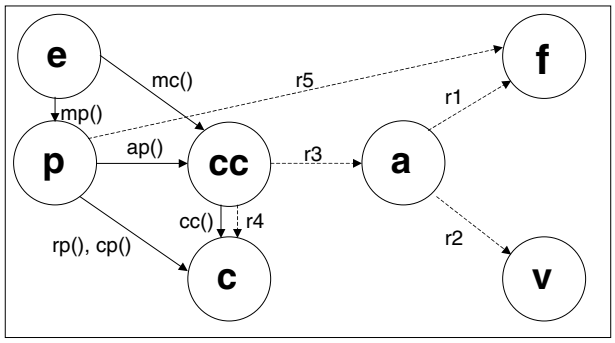

Figure 1: Dynamic evolution of the state of a commitment object.

In the following sections commitment objects are represented with the following notation:

$$C_{id}(state, debtor, creditor, content|condition).$$

For example:

- $C_i(a, a, b, P)$ is an active commitment by a, relative to b, that P will be satisfied.
- $C_i(cc, a, b, P|Q)$ is a conditional commitment by a, relative to b, that if Q is satisfied then also P will be satisfied.

3.1.2 Public Methods

We shall now describe the public methods that can be used to manipulate commitment objects. We assume that when a commitment object is declared, the constructor of the class allocates a memory block and creates an empty commitment object $C_i(e)$. In our notation the invocation of a method of the commitment object is represented by the name of the object followed by a dot and by the name of the method with its parameter list. For each method we describe the precondition and the result for its invocation. The precondition shows in which state an object must be in order that the method can be invoked on it and specifies also whether the agent that invokes the method, has to be the debtor or the creditor of the commitment object.

There are two methods to *make* a new commitment object, that is to initialize its fields to a specific value. The methods are:

- Make (conditional) commitment, $mc()$: agent a (the debtor of the new commitment i can invoke

  $$C_i(e).mc(a, b, P, Q) \rightarrow C_i(cc, a, b, P|Q).$$

  To create an active commitment it is enough to create a conditional commitment with a condition that is already true; this becomes an active commitment thanks to update rule 3 (see Table 1).
- Make precommitment, $mp()$: agent a (the creditor of the new precommitment i can invoke

  $$C_i(e).mp(a, b, P, Q) \rightarrow C_i(p, b, a, P|Q).$$

The accept precommitment method, $ap()$, changes the state of a commitment object from precommitment to conditional commitment (or to active commitment if the condition is already true, see update rule 3, Table 1). The accept method requires the existence of the precommitment object, and can be performed by agent a, the debtor of the commitment i, by invoking

$$C_i(p, a, b, P|Q).ap() \rightarrow C_i(cc, a, b, P|Q).$$

There are three methods to *cancel* an existing commitment object, that is to set its state to cancelled. The methods are:

- Cancel (conditional) commitment, $cc()$: agent a, the creditor of commitment i can invoke

  $$C_i(cc, b, a, P|Q).cc() \rightarrow C_i(c, b, a, P|Q).$$
- Cancel precommitment, $cp()$: agent a, the creditor of precommitment i, can invoke

  $$C_i(p, b, a, P|Q).cp() \rightarrow C_i(c, b, a, P|Q).$$
- Reject precommitment, $rp()$: agent a, the debtor of precommitment i, can invoke

  $$C_i(p, a, b, P|Q).rp() \rightarrow C_i(c, b, a, P|Q).$$

3.2 The temporal proposition class

As previously mentioned, an instance of the temporal proposition class can be assigned to the content or condition field of a commitment object. We refer to temporal proposition objects by a capital letter (P, Q,...). We define here only the private fields of the class; we do not define the methods to get field values, but we assume they exist.

3.2.1 Fields

Statement is a sentence in a suitable formal language, it may state that:

- A state of affairs (in the application domain) holds.
- An action (see Section 3.3) has been performed.
- A commitment holds. In this case the temporal proposition object becomes true when a commitment object, corresponding to this sentence, is actually created. This third case allows us to express conditional commitments, which become active only if another specific commitment object is created.

A Boolean combination of statements is a statement.

Time Interval expresses the time period to which the sentence in the statement field (see above) is referred.

Mode expresses the temporal qualification of the statement, which can be required to be true for a whole time interval ($\forall$) or to become true during the time interval ($\exists$).

State can be true (1), false (0) or undefined ($\perp$). It is up to a "notifier", connected with the meaning of the sentence to change the state from undefined to true or false, according to the following rules:

- If the mode is '$\forall$' the notifier sets the state to false if the sentence becomes false at any point of the time interval; otherwise the notifier sets the state to true when the time interval expires.

Table 1: Update Rules

event	action	rule
$P.state() = 1$	$C_i(a,a,b,P) \rightarrow C_i(f,a,b,P)$	1
$P.state() = 0$	$C_i(a,a,b,P) \rightarrow C_i(v,a,b,P)$	2
$Q.state() = 1$	$C_i(cc,a,b,P\|Q) \rightarrow C_i(a,a,b,P)$	3
$Q.state() = 0$	$C_i(cc,a,b,P\|Q) \rightarrow C_i(c,a,b,P\|Q)$	4
$P.state() = 1$	$C_i(p,a,b,P) \rightarrow C_i(f,a,b,P)$	5

- If the mode is '$\exists$' the notifier sets the state to true if the sentence becomes true at any point of the time interval; otherwise the notifier sets the state to false when the time interval expires.

It is important to note that when the state changes from undefined to true or false it can not change anymore. We denote with the symbol T a constant temporal proposition object with an empty statement and a state that is true.

3.3 Actions

An agent can perform actions in the environment. The set of executable actions includes:

- all methods of the commitment class and the actions that can be defined using those methods;
- all actions in a set of *application domain actions*. For example, in an interaction system used to negotiate some commodities, "pay x\$ to a" and "deliver product g to a" can be application domain actions.

3.4 Update Rules

When a temporal proposition object, associated with the content or condition field of a commitment object, changes its state from undefined to true or false, it is necessary to update the state of the commitment object. We shall formalize the event driven routines that are automatically invoked using the update rules described in Table 1. In the first column there is the event that triggers the rule, while the associated state transition is described in the second column. The method *state()* is a method of the temporal proposition class that returns the state of a temporal proposition object. In particular in Rule 5 the temporal proposition P describes an action performed by the debtor of the commitment. This rule is necessary to cover situations when an agent performs the requested action without accepting the request explicitly. A possible solution to make explicit the connection between the performance of an action and the request to perform that action is through an explicit reference made by the actor to the associated request.

4. DEFINITION OF MAIN SPEECH ACTS

In this section we use the previously defined operations on commitment objects to express the meaning of some important speech acts, especially the ones common used in multi agent interaction in open environments. In our analysis, we follow the taxonomy by John Searle [9] and classify illocutionary acts into five categories: declarations, assertives, commissives, directives and expressives. Here we do not treat expressives, because we think they are not relevant for artificial agent communication, and declarations, because they require an analysis that lies beyond the scope of this paper.

4.1 Assertives

The goal of an assertive act is to commit the speaker relative to the hearer to the truth of what is asserted. In human language, the simplest type of assertive act is *asserting*. However, we conform here to the practice, common in the ACL field, of considering *informing* as the prototypical assertive act.

inform. This act is used when an agent a wants to inform agent b about P, for example P can be "from now agent a is on-line" or "my name is ...". In a commitment-based approach, an act of informing can be defined as follows:

$$inform(a,b,P) =_{def} C_i(e).mc(a,b,P,T).$$

informIf. This is an abstract act, that is an agent can not actually do an informIf act but performing one of two specific inform acts is equivalent as performing an informIf act. An agent does an informIf act when it informs another agent about the truth value, either true or false, of a proposition. Let P be a temporal proposition object with sentence φ in its statement field. We shall call $P^{\neg}$ the temporal proposition object that has in the statement field the negation of the sentence φ. Assuming that the symbol "$\vee$" indicates the execution of one of two acts, the informIf act can be defined as follows

$$informIf(a,b,P) =_{def}$$
$$inform(a,b,P) \vee inform(a,b,P^{\neg}).$$

This abstract act will be used in the following sections to define questions.

informRef. This is another abstract act. We shall call $S(x)$ a temporal proposition object whose statement is a sentence $\varphi(x)$ containing a single free variable x. An agent actually does an informRef act about a given temporal proposition object $S(x)$ when it performs an inform act about a temporal proposition object $S(c)$, where $S(c)$ is obtained by $S(x)$ substituting variable x with constant c.

$$informRef(a,b,S(x)) =_{def} inform(a,b,S(c))$$

Also this abstract act will be used in the following sections to define questions.

4.2 Commissives

The point of a commissive act is to commit the debtor, relative to the creditor, to the execution of an action of a given type within a given interval of time. Here we define our basic commissive act, promising.

promise. An agent a promises to agent b that a will bring about P within a time interval in the future, for example P can be "I will visit you tomorrow".

$$promise(a,b,P) =_{def} C_i(e).mc(a,b,P,T)$$

It appears from this definition that there is no difference between agent a inform agent b that a will do an action in the future, and agent a promising to agent b that a will do an action. In Searle's speech act theory the difference between assertives and commissives is dealt with in terms of a property called *direction of fit*. Direction of fit may be relevant in a commitment-based approach, especially because it is likely to influence the management of commitment violations. However, at the present stage of development we do not include this aspect in our operational specification.

conditionalPromise. An agent a performs a conditional promise to agent b if it commits to bringing about P (within

a given time interval) if condition Q becomes true (within a given time interval). Agent b may or may not have the responsibility for condition Q to become true. An example where the condition does not depend on the hearer is, "If it rains tomorrow, I shall give you 5$ the day after tomorrow". An example where the condition depends on the hearer is, "If you give me this book within time t_1, I will give you 5$ within time t_2", with $t_1 < t_2$.

$$conditionalPromise(a, b, P, Q) =_{def} C_i(e).mc(a, b, P, Q)$$

4.3 Directives

The point of a directive act is to get the hearer to perform an action within an interval of time. As we already pointed out, an agent can not directly commit another agent to something; therefore, to define directives we use the concept of precommitment.

request. Our basic directive act is the request. A request by agent a to agent b to bring about P is defined as:

$$request(a, b, P) =_{def} C_i(e).mp(a, b, P, T)$$

Some requests can be satisfied immediately, like for example "Please close the door". To this kind of request the answer may be the execution of the requested action or a rejection. If the debtor of the precommitment immediately executes the action requested, update rule 5 is activated and the state of the commitment object changes from precommitment to fulfilled. On the contrary, the precommitment is cancelled if the request is rejected. In any case, we assume that a precommitment is automatically cancelled after a predefined time-out has expired. There are also request to do actions that cannot be immediately executed, for example "Please send me a copy of your book". Here the hearer (i.e., the debtor of the precommitment) may react by *accepting* or *rejecting* the request. If the debtor of the precommitment accepts it, then it is committed to do the action, otherwise it is not. To summarize the different possibilities, the hearer of a request can react in three different ways: it can perform the action, accept the request or reject the request. If the time-out has elapsed, the precommitment is cancelled.

conditionalRequest. It is also possible that the goal of the speaker is to get the hearer to accept a conditional commitment. We define conditional requests using a precommitment that has a condition. An example of this kind of request can be, "If it rains, can you give me a lift?". It is important to note that the realization of the condition is not in the power of the speaker. The request from agent a to agent b to bring about P within a certain interval of time if condition Q is satisfied is defined as:

$$conditionalRequest(a, b, P, Q) =_{def} C_i(e).mp(a, b, P, Q).$$

4.3.1 Accept and reject

The debtor of a precommitment can accept or reject it. The act of accepting a precommitment transforms it in an active commitment or in a conditional commitment, while the act of rejecting a precommitment has the effect of cancelling it.

$$accept(b, C_i(p, b, a, P|Q)) =_{def} C_i(p, b, a, P|Q).ap()$$
$$reject(b, C_i(p, b, a, P|Q)) =_{def} C_i(p, b, a, P|Q).rp()$$

4.3.2 Question

A question act is a particular directive act, by which the speaker tries to get the hearer to commit to the truth of a certain proposition. A question can be seen as a *request to inform* about something; therefore we define questions using the assertive acts previously defined. We shall analyze two type of questions.

yes/noQuestion. Yes/no questions are used to get the hearer to commit to the truth value of a given proposition. A yes/no question can be defined using the informIf act:

$$yes/noQuestion(a, b, S)) =_{def} request(a, b, P)$$
$$where \;\; P.statement() = informIf(b, a, S)$$

whQuestion. WhQuestion acts are used to get the hearer to identify an individual having a given property. The speaker provides a "frame" for the answer, and the hearer has to fill it with the correct piece of information. For example, a whQuestion act may be: "What is your name?", that has the same meaning as "Please tell me that your name is x" where x is a variable to which the hearer has to assign a constant value. A whQuestion act can be defined using the InformRef act.

$$whQuestion(a, b, S(x))) =_{def} request(a, b, P)$$
$$where \;\; P.statement() = informRef(b, a, S(x))$$

4.4 Proposals

A proposal is a conjunction of a conditional directive and a conditional commissive. More precisely, it is the conjunction of a conditionalRequest and a conditionalPromise. For example, we can analyze the following proposal made by agent a to agent b, "Will you give me good g within time t_2 if I give you x\$ within time t_1?", with $t_1 < t_2$. Note that a proposal is similar to a conditional request. The crucial difference is in the condition, because in proposals the realization of the condition depends on the speaker. A proposal can be defined as the execution of two actions, as indicated by the symbol "$\wedge$".

$$propose(a, b, P, Q) =_{def}$$
$$conditionalRequest(a, b, P, Q) \;\wedge$$
$$conditionalPromise(a, b, Q, S)$$
$$where \;\; S.statement() = C_i(cc, b, a, P|Q)$$

It is important to put in evidence that the conditionalPromise is related to the conditionalRequest through the common proposition object Q. In electronic commerce applications, for example in electronic auctions, the *bid* of a buyer is equivalent to a proposal as defined above.

5. A SAMPLE APPLICATION

In this section we formalize a standard interaction protocol within the previously defined framework. To deal with a significant example, we analyze a conversational protocol widely used in electronic commerce applications, in particular in some types of auctions: the protocol of proposals. The example shows:

- how it is possible to use commitment-based definitions of speech acts to express the meaning of the various messages exchanged in a negotiation;
- the dynamic evolution of the system states during an interaction;
- the use of the commitments to evaluate the *soundness* of the protocol.

A protocol is based on the set of the speech acts previously defined as operations on commitment objects. It is described by an *interaction diagram*, that is, a graph whose nodes represent system states, and whose edges represent certain types of state transitions. In an interaction diagram, state transitions correspond either to speech acts performed by the interacting agents, or to environmental events strictly related to the interaction.

To each state, a *content* is associated, consisting of commitment objects and temporal proposition objects holding at that particular state. Such objects are computed in the following way. The diagram has a distinguished *initial state* (*start*), whose content is a set of empty commitments. If state s_j is reached from state s_i by a speech act, then the content of state s_i must satisfy the preconditions of the speech act, and the content of s_j is obtained by modifying the content of s_i according to the definition of the speech act. If on the contrary state s_j is reached from state s_i by an environmental event, then the content of s_j is obtained by modifying the content of s_i according to the relevant update rule. States with no outgoing edges are classified as *final*; when a final state is reached, the interaction ends.

An arbitrary interaction diagram need not, in general, describe a "sensible" interaction. To do so, an interaction diagram must satisfy a number of *soundness conditions*. In particular:

- every state of the interaction diagram must be reachable from the start state;
- there must be at least a final state;
- all commitments included in the content of a final states must be empty, fulfilled or violated.

In special cases, there may be further soundness conditions. For example, many types of interaction include a first phase of *negotiation* and a second phase of *execution*. The negotiation phase has the start state, and ends in one or more *contract state*, whose content includes only commitments that are either active or conditional. From a contract state it is possible to enter the execution phase, that ends into final states as described above. The example reported below is a simple case of negotiation/execution interaction.

5.0.1 Proposal equivalent to a bid

The example shows the dynamic evolution of the system states when a propose act is performed by agent a addressing agent b. Figure 2 reports the interaction diagram, while the content of each state is described in Table 2. States $s_0, ..., s_3$ represent the negotiation phase of the interaction protocol; in particular, the content of state s_3 represents the contract between the two agents. States $s_4, ..., s_7$ represent the execution of the contract. Final states are marked by double blocks.

It is easy to check that the proposal protocol satisfies all previously described soundness conditions.

6. CONCLUSIONS

In this paper we have presented an operational definition of a commitment-based semantics for agent speech acts, and shown with an example that this semantics can be used to specify interaction protocols. In particular, we believe that an interesting contribution of the paper is the definition of a proposal as a combination of a conditional request and a conditional promise. We also show, as an example, how an

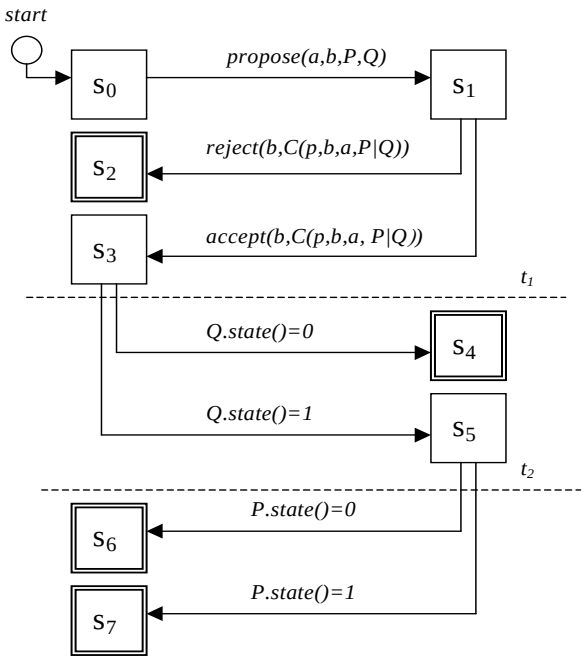

Figure 2: Interaction diagram of the proposal protocol (see Table 2 for state contents).

Table 2: Contents associated to the states of the interaction diagram of the proposal protocol.

$P(b \text{ do } \beta, now \dots t_2, \exists, \bot)$ $Q(a \text{ do } \alpha, now \dots t_1, \exists, \bot)$ $t_1 < t_2$

s	reason of the action	content
s_0	*start*	$C_i(e)$ $C_j(e)$
s_1	$propose(a, b, P, Q)$	$C_i(p, b, a, P\|Q)$ $S.statement() = C_i(cc, b, a, P\|Q))$ $C_j(cc, a, b, Q\|S)$
s_2	$reject(b, C_i(p, b, a, P\|Q))$ $notifier$ rule 4	$C_i(e)$ $S.state() = 0$ $C_j(e)$
s_3	$accept(b, C_i(p, b, a, P\|Q)$ $notifier$ rule 3	$C_i(cc, b, a, P\|Q)$ $S.state() = 1$ $C_j(a, a, b, Q)$
s_4	$Q.state = 0$, rule 4 rule 2	$C_i(e)$ $C_j(v, a, b, Q)$
s_5	$Q.state = 1$, rule 3 rule 1	$C_i(a, b, a, P)$ $C_j(f, a, b, Q)$
s_6	$P.state = 0$, rule 2	$C_i(v, b, a, P)$ $C_j(f, a, b, Q)$
s_7	$P.state = 1$, rule 1	$C_i(f, b, a, P)$ $C_j(f, a, b, Q)$

interaction between two agents may be carried out when a proposal is made.

In the presented model temporal aspects are only sketched, we plan to develop this point in the future. In future studies we plan also to extend the model in order to cover some other speech acts, in particular orders and commands treated in hierarchical contexts. Moreover the expressiveness of the model may be increased with the addition of new methods to the commitment class to let interacting agents to negotiate the content and the condition of a commitment object.

Our approach is strongly related to the one proposed by Yolum and Singh [13]. However, between the two approaches there are some significant differences. First, our proposal provides a more complete account of how different types of speech acts can be defined in terms of operations on commitments. Second, and more important, we also provide a commitment-based analysis of directive speech acts, like requests, relying on the notion of precommitment.

An important point, that we have only briefly introduced, is the concept of soundness of interaction protocols. In the paper we have stated a number of soundness conditions relying only on intuition and without any deep theoretical justification. We believe, however, that it is possible to derive such conditions from a set of general interaction principles expressed in terms of *conversational commitments*, that is, of meta-level commitments that regulate conversational interactions among agents. A similar problem is addressed in [11]. To gain a more concrete idea of this concept, consider the *time-out* field of the commitment class. This field represents the time limit for the debtor of a precommitment to accept, fulfill, or reject it. In fact, it is possible to view the time-out of a precommitment as the *deadline* of another commitment, namely the conversational commitment that every agent makes when it decides to take part in an interaction, and that binds the agent to react to the speech acts performed by the other participants. The systematic development of this concept is an aim of our further research.

Finally, we would like to stress that while we consider the operational definition of commitment a very important step in view of practical applications, a thorough treatment of commitment-based semantics requires the development of a full *logic of commitment*. This important theoretical aspect is currently under development.

7. REFERENCES

[1] J. L. Austin. *How to Do Things With Words*. Oxford University Press, Oxford, 1962.

[2] C. Castelfranchi. Commitments: from individual intentions to groups and organizations. In V. Lesser, editor, *Proceedings of the First International Conference on Multi–Agent Systems*, pages 41–48, San Francisco, CA, 1995. MIT Press.

[3] M. Colombetti. A commitment-based approach to agent speech acts and conversations. In *Proc. Workshop on Agent Languages and Communication Policies, 4th International Conference on Autonomous Agents (Agents 2000), Barcelona (E)*, pages 21–29, 2000.

[4] M. Colombetti. A language for artificial agents. *Studies in Communication Sciences*, 1(12):1–32, 2001.

[5] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. M. Bradshaw, editor, *Software Agents*, chapter 14, pages 291–316. AAAI Press / The MIT Press, 1997.

[6] FIPA (2000). *Agent Communication Language. FIPA 2000 Specification*. Foundation for Intelligent Phisical Agents, http://www.fipa.org, 2000.

[7] J. Pitt and A. Mamdani. A protocol-based semantics for an agent communication language. In D. Thomas, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99-Vol1)*, pages 486–491, S.F., July 31–Aug. 6 1999. Morgan Kaufmann Publishers.

[8] J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, United Kingdom, 1969.

[9] J. R. Searle. A taxonomy of illocutionary acts. In K. Gunderson, editor, *Language, Mind, and Knowledge. Minnesota Studies in the Philosophy of Science, Vol. 7*, pages 344–369. University of Minnesota Press, Minneapolis, Minnesota, 1975.

[10] M. P. Singh. Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12):40–47, 1998.

[11] D. R. Traum and J. F. Allen. Discourse obligations in dialogue processing. In J. Pustejovsky, editor, *Proceedings of the Thirty-Second Meeting of the Association for Computational Linguistics*, pages 1–8, San Francisco, 1994. Association for Computational Linguistics, Morgan Kaufmann.

[12] M. Venkatraman and M. Singh. Verifying Compliance with Commitment Protocols. *Autonomous Agents and Multi-Agent Systems*, 2(3):217–236, 1999. Special Issue on Coordination Mechanisms and Patterns for Web Agents.

[13] M. P. Yolum and P. Singh. Synthesizing finite state machines for communication protocols. Technical Report TR-2001-06, Department of Computer Science, North Carolina State University, June 27 2001.