Engineering Issues in Inter-Agent Dialogues

Nikos Karacapilidis[1] and **Pavlos Moraïtis**[2]

Abstract. This paper presents a logical framework for modeling of complex dialogues between intelligent and autonomous agents. Our overall approach builds on the assumption that an agent is composed of a set of modules, each of them conveying the appropriate knowledge to carry out a certain dialogue type, such as deliberation, negotiation, persuasion, etc. Much attention has been paid in keeping our framework as operational as possible, in that the architecture of agents and their conversational protocol are thoroughly interrelated. Due to the proposed knowledge structure, set of application-independent rules (called dialogue policies) and the combination of backward and forward reasoning, the framework can generate automatic dialogues between agents.

1 INTRODUCTION

Research on agent communication attracts increasing interest in the last few years, while focuses on aspects such as conversation protocols and formal frames supporting different dialogue types. More specifically, several interesting works addressing the particularities of persuasion dialogues [14], negotiation dialogues [1, 7, 12, 13], deliberation dialogues [3], and combinations of dialogues [5, 6, 9] have been already proposed.

This paper engineers a set of formal and operational issues arising in a framework for inter-agent dialogues. Having first defined the concepts of *communication performative* and *dialogue context*, it proceeds by discussing procedures and rules related to the automation of dialogues. Much attention is paid to the definition of a set of application-independent rules, namely *dialogue policies*, which are exploited by a reasoning mechanism and serve the automated generation of the appropriate illocutionary acts, as well as the initiation and termination of a dialogue. The above concept is similar to that of dialogue constraints [12]; however, our dialogue policies are associated with the specific profile of an agent, thus characterizing his personality and behavior.

The main contribution of this work is in the seamless integration of the conversation model and agents architecture (both are fully implemented). In fact, agents in our framework operate combining two types of reasoning, namely *backward* (aiming at satisfying the goal generated upon the reception of a communication performative), and *forward* (building the appropriate answers to the performatives received), thus regulating the continuity of dialogues. In addition, our approach is based on the assumption that an agent is composed of a set of *modules*, each of them being responsible for a particular feature of the agent's overall behavior. Such features may concern abilities such as information seeking, deliberation, negotiation, etc. [14] allowing each module to perform the associated dialogue. Moves from one dialogue type to another are performed through the interaction of the different modules an agent consists of.

The remainder of the paper is structured as follows: Section 2 presents the proposed structure of an agent focusing on the representation of the knowledge conveyed, while Section 3 describes the conversation protocol defining the concepts of performative, dialogue policy and dialogue context. Section 4 gives some illustrative examples demonstrating the abilities of our approach, while Section 5 sketches the execution cycle of an agent's module. Finally, Section 6 comments on related work and outlines future work directions.

2 THE AGENTS STRUCTURE

Let Ag be the set of agents involved in an agent-based system. We assume that each agent $x \in Ag$ is composed of a set of modules Δ_x that implement its overall behavior. More specifically, each module δ_x ($\Delta_x = \cup \delta_x$) is responsible for a specific aspect of the agent's behavior (it may correspond to abilities such as deliberate, negotiate, cooperate, information seeking, etc.), where the overall behavior of an agent is the result of the interaction among the different elements of Δ_x. For instance, the role of an agent's deliberation module is to collaborate with its peers in order to decide a sequence of actions in some situations, while the role of its negotiation module is to apply the appropriate negotiation protocol in order to negotiate with another agent about a specific topic or goal. The above assumption, being in line with the work presented in [11], makes our approach different compared to existing work on developing frameworks for conversational agents. It should be made clear here that one may build an agent according to his/her particular interests in a specific application; that is, an agent x may be composed of just a negotiation module, while another one y may also include a deliberation and an information seeking module. Our architecture permits multiple parallel agent dialogues of different type (each agent has only one instance of the module types mentioned above; thus, he cannot be engaged in multiple parallel dialogues of the same type).

Each of the above modules is triggered whenever it is necessary to play the specific role it is conceived for, thus performing a dialogue corresponding to its "area of expertise". The idea is that the reaction of an agent to an input received (i.e., a message from another agent), or his global action towards achieving a goal, is based on a sequence of actions triggered in one or more (possibly all) of his modules. In other words, each agent's module is associated with a certain part of the overall dialogue. We also assume that all messages exchanged between two conversational agents pass through their *communication modules*. That is, the only constraint we impose in the agents' architecture is that an agent has to interact with his environment through the above module (no other restriction holds for the interaction between the other modules).

Finally, in order to serve its role, we assume that each module δ_x is equipped with a knowledge base $K(\delta_x)$. The content of this knowledge base may be application-specific or application-independent as well as module-specific or module-independent. However, in any case, it is related to the role of the particular module. The knowledge conveyed is expressed in a declarative way (first-order logic), as described below:

Definition 1. A knowledge base is a tuple <F, G, A, solver, DP, PR, messenger, RF, D>, where (see Fig. 1):

- F contains *application-specific knowledge (facts)* related to the role of the module (e.g., for a deliberation module, such

[1] Industrial Management Lab, MEAD, University of Patras, 26504 Rio Patras, Greece, nikos@mech.upatras.gr

[2] Dept. of Computer Science, University of Cyprus, 75 Kallipoleos str., Nicosia, Cyprus, moraitis@ucy.ac.cy

knowledge may concern the agent's environment) and the specific topics.

- G is the *goal* to be achieved (represented by sentences).
- A is the set of possible *actions* (represented by *if-then* rules).
- solver is an *application-independent inference engine* that exploits facts and actions to reach a goal. It is activated whenever a new goal G' replaces (the existing) goal G. It is based on a *backward reasoning* mechanism.
- DP is a set of application-independent knowledge, namely *dialogue policies*, represented by *if-then* rules (see next Section). The messenger uses these policies to regulate the dialogues.
- PR is a set of *preference relations* $\succ^{pr}$ on the set of F and on the set of A.
- messenger is an *application-independent inference engine* that filters the received messages and permanently consults the existing dialogue policies and preference relations. It exploits a *forward reasoning* mechanism.
- RF is a list of the *reasons* (facts or actions) leading to the *failure* of the current goal.
- D contains the messages exchanged during the current dialogue (messages exchanged between the modules of two different agents through their communication modules and/or, in case of embedded dialogues, between two modules of the same agent directly). It is implemented as a queue that is emptied after the end of each dialogue.

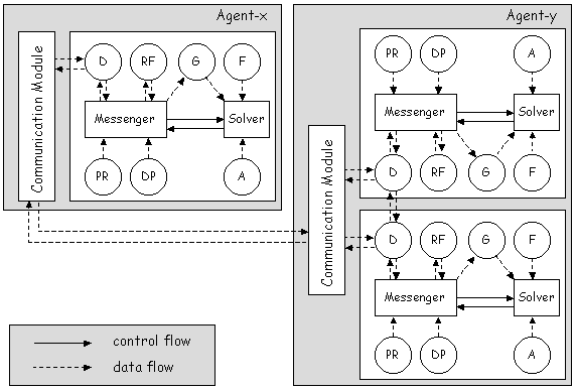

Figure 1: *Structure of agents.*

It should be noted here that the agent's knowledge bases may change over time due to the outcome of a negotiation, argumentation or persuasion dialogue.

3 THE CONVERSATION PROTOCOL

Conversation between agents is based on dialogues. We make the simplifying assumptions that only two agents may participate in a dialogue and that an agent's module may be involved in just one dialogue at each time. A dialogue between agents can be a complex process, taking place through the exchange of messages. Each message conveys certain semantics in order to be appropriately handled by an agent. A dialogue is always initiated in order to achieve a goal and, according to the nature of this goal (i.e. perform a task, persuade the validity of a statement, etc.), it installs the appropriate context (i.e. deliberation process, negotiation process, etc.). In a dialogue context, each time an agent receives a message, it has to know immediately which reasoning procedure (that is, which module) it must be activated in order to set up the most appropriate answer (or action) to the message received.

Definition 2. A *message* is an instance of a schema of the form Msg=(id, P), where P declares the *performative* (dialogue primitive) conveyed. In our framework, it is $P=S(x, y, \sigma, T)$, where:

- id is the message's identification number;
- S is an illocutionary act belonging to the set {propose, accept, request, assert, refuse, challenge, reject} (note that this is the set implemented in the current stage of our work; obviously, it can be altered according to the particular dialogue setting);
- x and y are the sender and the receiver of the performative, respectively;
- σ is the subject (i.e., body) of the performative, which may take one of the following forms:
  - a tuple σ = <sentence [support]> where support consists of elements (facts, actions, etc.) expressing arguments supporting sentences. When no support is available (or necessary to be explicitly mentioned), its value is $\emptyset$;
  - a dialogue context structure DC (see below);
  - $\emptyset$, meaning *"nothing to say"*.
- T is the time when the performative is uttered (times are actually timestamps of the related transaction).

In fact, the first of the forms proposed for σ may express any message content, provided that it respects first-order logic representation.

Definition 3. For an agent $x \in Ag$, a *dialogue policy* is an *if-then* rule of the form $P(x, y, \sigma, T) \wedge C \Rightarrow P'(y, x, \sigma', T+1)$, where:

- $P(x, y, \sigma, T)$ is a performative uttered at time T, $P'(y, x, \sigma', T+1)$ is a performative sent at time (T+1) from the receiver of $P(x, y, \sigma, T)$ to its utterer, and σ (σ') is the subject of the performative, as described above (the subject of P' is not always the same with that of P). The above concept is similar to that of dialogue constraints [12], which however correspond to integrity constraints in an abductive logic programming framework. In the rest of this paper, the part $(P(x, y, \sigma, T) \wedge C)$ is referred to as body(dp) and $P'(y, x, \sigma', T+1)$ as head(dp).
- C, hereafter referred to as *condition*.

It should be made clear here that our dialogue policies actually encode the personality of an agent (such as "co-operative", "hard-nosed", "self-interested", etc.). The idea is that when a module of an agent receives a message related to a subset of the defined dialogue policies (see DP1, DP2, DP3, DP4, DP5 below), its subject σ is considered as a goal G^{σ}. The operation of the solver (which uses backward reasoning exploiting the *if-then* rules of the set A), corresponds to the reduction of G to sub-goals, which in turn correspond to the *if* parts (or premises) of the triggered rules. The satisfaction (or not) of these rules defines what DP will be triggered and consequently what is the condition C to be checked in order to choose the message to be sent. Otherwise, C depends on the type of the received message (see DP6, DP7, DP8, DP9). In other words, the dialogue policy is a procedure of entailment that defines what is the next message to be sent by an agent y, after the reception of a specific message coming from another agent x. As defined above, a dialogue policy is a part of the knowledge base of each module of an agent; the module is triggered appropriately whenever the knowledge it contains is sufficient for action at a certain time instance. The definition of the agent's architecture as a combination of different modules and the presence of dialogue policies in the knowledge base of each of them sets an important difference of our approach to the one proposed in [12]. More specifically, their proposed agent's architecture is the combination of a *communication layer*, a *planning module* and a *reasoning module*, where the last is being in charge of activating a sequence of dialogues (and therefore equipped with the related knowledge) no matter which is the type of the undertaken dialogue (i.e. deliberation, negotiation, persuasion, etc.) [14].

In order to regulate the interchange of messages used in our approach, we have defined a set SoP of the allowed sequences of performatives (expressed through dialogue policies). It is: SoP={

request → assert, request → refuse, propose → accept, propose → refuse, propose → propose, refuse → challenge, challenge → assert, assert → accept, assert → reject}. The dialogue policies corresponding to a (rather usual) profile are formally presented below (note that condition C appears within the square brackets):

DP1: request (x, y, σ, T) ∧ [K(δ_y) ⊢ G^{σ}] ⇒ assert (y, x, σ, T+1)

DP2: request (x, y, σ, T) ∧ [K(δ_y) ⊬ G^{σ}] ⇒ refuse (y, x, σ, T+1)

DP3: propose (x, y, σ, T) ∧ [K(δ_y) ⊢ G^{σ}] ⇒ accept (y, x, σ, T+1)

DP4: propose (x, y, σ, T) ∧ [K(δ_y) ⊬ G^{σ} ∧ {∃ σ' ∈ K(δ_y) | (σ' $>^{pr}$ σ) ∧ (K(δ_y) ⊢ $G^{\sigma'}$)}] ⇒ propose (y, x, σ', T+1)

The first three dialogue policies are rather straightforward. The condition imposed in DP4 means that if K(δ_y) does not entail G^{σ} (i.e., the goal associated to the subject σ of the received performative), it is checked whether there exists another σ' belonging to K(δ_y) along with a preference in PR stating that σ' is preferable than σ, such that $G^{\sigma'}$ can be entailed by K(δ_y).

DP5: propose (x, y, σ, T) ∧ [K(δ_y) ⊬ G^{σ} ∧ {∄ σ' ∈ K(δ_y) | (σ' $>^{pr}$ σ) ∧ (K(δ_y) ⊢ $G^{\sigma'}$)}] ⇒ refuse (y, x, σ, T+1)

The condition above is similar to the one of DP4, with the difference that in this case there is not a σ' belonging to K(δ_y) along with a preference σ' $>^{pr}$ σ, such that $G^{\sigma'}$ can be entailed by K(δ_y).

DP6: refuse (x, y, σ, T) ∧ [support (σ) = ∅] ⇒ challenge (y, x, σ, T+1)

The meaning of the above is that an unsupported refusal, sent from agent x to agent y, triggers a challenge act from y (which actually asks x to justify his decision).

DP7: challenge (x, y, σ, T) ∧ [∃ reason ∈ RF(y) | reason ⊢ (¬σ)] ⇒ assert (y, x, σ, T+1)

The condition above actually checks RF to verify whether there exists a reason of failure of the goal associated to the subject σ (i.e., a fact or an action that contradicts σ); if yes, the reason found is sent back to the utterer of the challenge act (as a support of σ).

DP8: assert (x, y, σ, T) ∧ [support (σ) ≠ ∅ ∧ {∄ support' ∈ K(δ_y) | support' ⊢ (¬support)}] ⇒ accept(y, x, σ, T+1)

DP9: assert (x, y, σ, T) ∧ [support (σ) ≠ ∅ ∧ {∃ support' ∈ K(δ_y) | support' ⊢ (¬support)}] ⇒ reject(y, x, σ, T+1)

The meaning of the condition of DP8 is that if a support', which contradicts support, cannot be found then y has to accept the assertion of x. Otherwise (in case that a support', which contradicts support, exists in the knowledge base of y), y will reject it (in fact, it will reject the support provided). In addition to the above, the following two policies (exploiting the *predicates* need and want_share) are used for the initiation of a dialogue:

DP10: need (x, r, goal) ∧ [¬have (x, r) ∧ have (y, r)] ⇒ request (x, y, give (y, x, r)), where r can be a certain resource.

DP11: have (x, r) ∧ [want_share (x, y, r)] ⇒ propose (x, y, r), where r can be a certain desire, goal, or resource.

Definition 4. Given that agents convey knowledge in their constituent modules, as described in the previous section, and inspired by the work presented in [9, 14], we define a *dialogue context* as a tuple (t, (τ, M)), where:

- t is the *type* of the dialogue (t ∈ {deliberation, negotiation, cooperation, argumentation, …});
- τ is the *topic* of the dialogue (i.e., what agents discuss about);
- M is the *medium* used for the dialogue, which may refer to messages exchanged between two conversational agents x and y either directly (denoted by Direct) or through a *mediator* z (denoted by med(z)), messages shared via a common memory w (such as a *blackboard*), etc.

Often, agents may get involved in a dialogue about what they will discuss in the sequel. For instance, during a deliberation dialogue between two agents x and y intending to decide which car to buy, agent x may initiate a new dialogue type in order to negotiate with his peer about the list of criteria to be considered in their decision. In this case, the initial dialogue context would be DC_k=(deliberate, (car_purchase, Direct)), where a new context DC_m=(negotiate, (criteria, Direct)) will be settled (embedded to the previous one) due to the proposal of x. The above is in agreement with one of the key features of the approach proposed in the work of Walton and Krabbe [14], assuming that dialogue types may be nested. In such cases, a new dialogue type is initiated by a specific message, whose subject is a dialogue context structure. Agents thus know in which dialogue type the messages exchanged in the sequel belong to.

Agents utter successively in a dialogue while the choice of the appropriate message to be sent at each time is based on the set of the dialogue policies residing in the associated module of each agent. This means that a dialogue of a certain type installs a specific framework of interaction (i.e., *dialogue context*) that triggers the appropriate module of each agent. More specifically, it is the type t of a dialogue context that triggers the appropriate module, provided that the topic τ belongs to the facts F of the knowledge base K(δ_x), where δ_x is the module of agent x that corresponds to the dialogue type t (for more details, see the second example in the next section). The rationale of the above is that an agent must have the appropriate module and be equipped with the necessary knowledge in order to be able to discuss about the specific topic (i.e., it would be rather surprising for a fish-market's seller agent to deliberate on weather forecasting with a meteorologist agent).

Definition 5. Adapting the approach followed in [12] to our framework, *normal termination of a dialogue* occurs when, given that an agent x utters a performative p_i, there exist no possible performative that y can utter after consulting the list of dialogues policies DP, that is, for all dp ∈ DP (note that a dp has been previously defined as body(dp) ⇒ head(dp)), it holds (K(δ_y) ∪ p_i) ⊬ body(dp). This definition will be extended in the future, to also capture abnormal termination cases (concerning communication failure between agents).

4 SOME EXAMPLES

In order to better describe the functionality of our model, we present below two examples (for clarity reasons, we will use the concept of performative instead that of a message to omit the associated message ids). The first example concerns a widely used scenario involving home-improvement agent (see for instance, [1, 7]). According to it, agent X has the intention of hanging a picture, knows how to do it by using a nail, but lacks the necessary nail. On the other hand, agent Y has the intention of hanging a mirror, knows how to do this, and has all the necessary resources. Among these resources, there is only one nail, which X would like to get. The dialogue taking place between X and Y is as follows:

X: Please give me a nail. Y: No. X: Why won't you give to me?
Y: Because I want to hang a mirror and for that I need a nail.
X: I understand.

Using our framework, the negotiation module of agent X has the following knowledge:

- F_X = {F_{X1}: ¬Have (X, NAIL), F_{X2}: Have (Y, NAIL)};
- A_X = {A_{X1}: ∀z, ∀goal, Have(X,z) ∧ Achieve (X,z,goal) ⇒ Need (X,z, goal)};
- G_X = achieve (X, NAIL, PICTURE_HANGED);
- DP = {DP10: ∀ y, ∀ r, ∀ goal, Need (X, r, goal) ∧ ¬Have (X, r) ∧ Have (y, r) ⇒ request (X, y, Give (y, X, r))}.

According to the above, agent X will send to agent Y the performative P_k = request (X, Y, <Give (Y, X, NAIL) [∅]>, T). On the other hand, the negotiation module of Y has in its own knowledge base the following:

- F_Y = {F_{Y1}: Have (Y, NAIL)};
- A_Y = {A_{Y1}: ∀x, ∀z, ∀goal, Have (Y,z) ∧ ¬Need (Y,z,goal) ⇒ Give (Y, x, z), A_{Y2}: ∀ z, ∀ goal, Have (Y, z) ∧ Achieve (Y, z, goal) ⇒ Need (Y, z, goal)};
- G_Y = Achieve (Y, NAIL, MIRROR_HANGED).

Having received the performative P_k, the messenger of Y detects that its illocutionary act is "request", it thus sets Give(Y, X, NAIL) as the goal G_Y to be obtained. The solver is then activated which, taking into account the above knowledge and following a backward reasoning mechanism (on A_{Y1} and A_{Y2}), concludes that G_Y cannot be achieved (see Fig. 2).

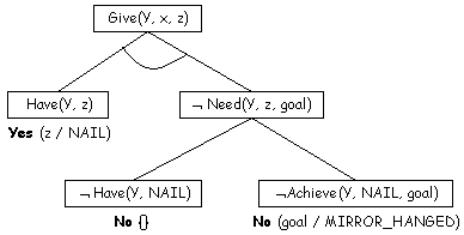

Figure 2: *Proof tree (first example)*

The messenger will be then triggered again which, by consulting the dialogue policy DP2, will generate the performative P_{k+1} = refuse (Y, X, ‹Give(Y, X, NAIL) [∅]›, T+1). Note that the reason leading to the failure of the goal is A_{Y2}, which has been put in RF_Y. Having received the performative P_{k+1}, the messenger of X detects that its illocutionary act is "refuse" and, due to the lack of support in P_{k+1} (support (Give(Y, X, NAIL)) = ∅) and DP6, sends the performative P_{k+2} = challenge (X, Y, ‹Give(Y, X, NAIL)›, T+2). Agent Y receives the above performative, his messenger is again activated, detects that it is a "challenge" message and, due to DP7, consults the list RF_Y and sends the performative P_{k+3} = assert (Y, X, ‹Give(Y, X, NAIL)›, [Have(Y, NAIL) ∧ Achieve(Y, NAIL, MIRROR_HANGED) ⇒ Need(Y, NAIL, MIRROR_HANGED)], T+3). Proceeding similarly, X receives P_{k+3}, his messenger detects that its illocutionary act is "assert" and, due to DP8, checks whether his knowledge base $K(\delta_X)$ contains any contradictory information. Since this is not the case in our example (see A_{X1}), X cannot further defeat the last assertion of Y, and sends the performative P_{k+4} = accept (X, Y, ∅, T+4). ■

As noted in the previous section, in order to set up a dialogue, an agent has to send a message conveying a dialogue context in its body. In this second example (see Fig. 3 illustrating the related inter-agent dialogue), consider an agent Z that wants to negotiate with his peer Y about a certain topic. The message request (Z, Y, ‹Negotiation(Topic:GOING OUT, Direct)›, 1) will actually trigger Y to check whether he is able to participate in such a dialogue, that is whether he has a negotiation module and the topic exists in his knowledge base. Assuming that the above hold, Y confirms its ability to participate in such a dialogue by sending the message accept (Y, Z, ‹Negotiation(Topic: GOING OUT, Direct)›, 2). Upon the receipt of this confirmation, assume that Z sends a message Msg_1 containing the performative propose (Z, Y, ‹Have_a_dinner(Z, Y, ESTIADES)›, 3) (for space reasons, we omit details concerning the construction of this message). Let Y having the following knowledge (in its related module):

- F_Y = {F_{Y1}: ¬Good_restaurant(ESTIADES), F_{Y2}: Good_film(AI), F_{Y3}: Friend(Z), F_{Y4}: Have(Y, TIME), F_{Y5}: ¬Good_film(LEGALLY BLONDE)};
- A_Y = {A_{Y1}: ∀ x, ∀ w, ∀ time, Have(Y, time) ∧ Good_restaurant(w) ∧ Friend(x) ⇒ Have_a_dinner(Y, x, w), A_{Y2}: ∀ x, ∀ z, ∀ time, Have(Y, time) ∧ Good_film(z) ∧ Friend(x) ⇒ Go_cinema(Y, x, z)}
- PR_Y = {PR_{Y1}: ∀ x, ∀ z, ∀ w, Go_cinema(Y, x, z) $>^{pr}$ Have_a_dinner(Y, x, w)}.

Having received Msg_1, the messenger of the corresponding module detects that its illocutionary act is "propose" and sets the subject of the message as the goal G to be obtained. That is, the current goal of agent Y is: Have_a_dinner(Z, Y, ESTIADES). In the sequel, the module's solver is triggered (as shown in Fig.1, solver is permanently consulting the sets A and F). Due to the action A_{Y1}, Have(Y, time) ∧ Good_restaurant(w) ∧ Friend(x) becomes now the new goal to be satisfied.

Following a *backward chaining* algorithm [10], this goal cannot be achieved (ESTIADES is not considered to be a good restaurant). Therefore, agent Y cannot answer positively to the proposal of Z. The fact ¬Good_restaurant(ESTIADES) is put in the list RF of the related knowledge base of Y, the idea being that Y can use this element as an *argument* supporting its future action(s) or decision(s). Since the current goal cannot be obtained, the messenger of Y is activated again (as in Fig.1, messenger is permanently consulting PR and DP). Due to the preference PR_{Y1} and the dialogue policy DP4, a new goal $G^{\sigma'}$ is now defined (corresponding to the last part of the condition C of DP4) and solver is once again activated in order to infer if this can be obtained. Easily, one can conclude that $G^{\sigma'}$ is satisfied since it is entailed by $K(\delta_Y)$ (due to F_{Y1}, F_{Y2} and A_{Y2}).

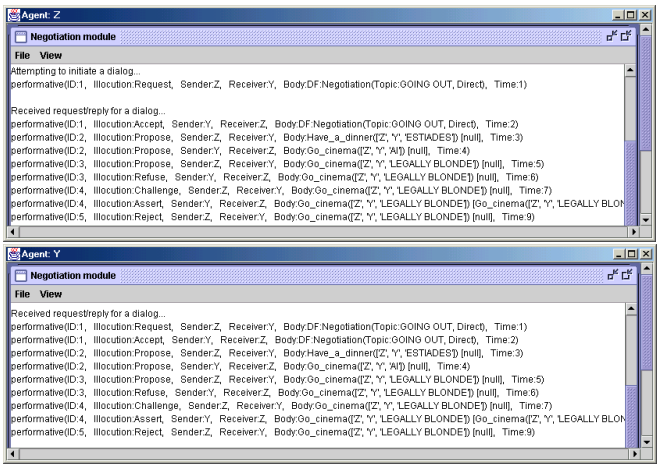

Figure 3: *Inter-agent dialogue (second example)*

DP4 actually enables agent Y to make a counter proposal to agent Z. According to the above, the answer of Y will be a message Msg_2 conveying the performative propose (Y, Z, ‹Go_cinema(Y, Z, AI)›, 4). To further continue this dialogue, assume that Z has the following knowledge (note that the items listed below are the ones needed to follow the example):

- F_Z = {F_{Z1}: Good_film(LEGALLY BLONDE), F_{Z2}: Friend(Y), F_{Z3}: Have(Z, TIME), F_{Z4}: ¬Good_film(AI)};
- A_Z = {A_{Z1}: ∀ x, ∀ y, ∀ time, Have(Z, time) ∧ Good_film(y) ∧ Friend(x) ⇒ Go_cinema(Z, x, y)}
- PR_Z = {PR_{Z1}: ∀x,∀y, Go_cinema(Z,x,LEGALLY BLONDE) $>^{pr}$ Go_cinema(Z, x, y}.

Exploiting DP4 again (and due to PR_{Z1}), agent Z makes another counter proposal: propose (Z, Y, ‹Go_cinema(Z, Y, LEGALLY BLONDE)›, 5). Having contradictory knowledge about this film (see F_{Y5}), Y sends the (unsupported) reply refuse (Y, Z, ‹Go_cinema(Z, Y, LEGALLY BLONDE)›, 6) (see DP5). Using DP6, Z may challenge the last reply; thus, he sends the performative challenge (Z, Y, ‹Go_cinema(Z, Y, LEGALLY BLONDE)›, 7). In turn, Y sends back to Z the reason leading to the failure of the last proposal (see DP7 and F_{Y5}). Since the support provided contradicts with F_{Z1}, agent Z will reject it. ■

5 THE EXECUTION CYCLE

It is generally admitted that in order to maintain the dynamic nature of a dialogue, the selection of the appropriate moves to be triggered should be based on rational and reactive reasoning mechanisms (see, for instance, [4]). Following our approach, the execution cycle of an agent's module combines elements of the

proposed structure of its knowledge base with issues related to the conversational protocol and reasoning mechanisms described in Section 3. Having defined the structure of agents, we sketch below the execution cycle of each such module (due to space reasons, we cannot provide a more detailed analysis of the algorithm).

```
begin
  for each incoming message (id, S (x, y, σ, T)) in D
    activate messenger;
    perform forward reasoning;
    illocutionary_act ← S;
    p ← S (x, y, σ, T);          /* p is a performative */
    if illocutionary_act ∈ {refuse, challenge, assert}
      read list DP;        /* DP is the list of dialogue policies */
      find dp_i where p belongs to body(dp_i)
        if K(δ_y) entails C     /*C is the condition of a dp */
          p' ← head(dp_i);
          new_message ← (id+1, p'); send new_message
        else exit
    end if
    if illocutionary_act ∈ {request, propose}
      G_δ ← σ;                  /* δ refers to the specific module */
      activate solver;
      perform backward reasoning;
      if G_δ is entailed
        read list DP;      /* performed by messenger */
        if illocutionary_act = request
          p' ← head(dp_1);
          new_message ← (id+1, p'); send new_message
        else                /* illocutionary_act = propose */
          p' ← head(dp_3);
          new_message ← (id+1, p'); send new_message
      else                  /* G_δ is not entailed */
        reason ← (reason of failure);
        put reason in RF;
        if illocutionary_act = request
          p' ← head(dp_2);
          new_message ← (id+1, p'); send new_message
        else                /* illocutionary_act = propose */
          if ∃ σ' ∈ K(δ_y) such that (σ' >^pr σ)
            if G^σ' is entailed    /* performed by solver*/
        /*all the following are performed by messenger*/
                    p' ← head(dp_4);
                    new_message ← (id+1, p');
                    send new_message
          else
              p' ← head(dp_5);
              new_message ← (id+1, p');
              send new_message
      terminate solver
    end if
    terminate messenger
  end for
end
```

6 DISCUSSION

This paper builds on previous related work to propose a formal and operational framework for dialogues between intelligent and autonomous agents. Issues addressed concern a modular agent implementation associated to the disparity of dialogue types in which an agent may get involved, a detailed knowledge structure together with the associated mechanisms for backward and forward reasoning, and the ubiquitous dialogue games.

Compared to previous work, our contribution has several advantages. First, having defined the illocutionary acts permitted, as well as combinations of these acts, the conversational model proposed is expressive enough to represent disparate (possibly embedded) types of dialogues in a unified way (we do not, however, claim that our model is more expressive than those described in [3, 6]). This has to be considered in parallel with our conception of agents' structure; that is, agents consist of a set of components, each of them being provided with an instance of the proposed conversation's model and being responsible to undertake a specific type of dialogue, related to its role in the overall agent's behavior (the type and the number of the involved modules depends on the application and the designer of the system). Second, due to the set of the dialogue policies defined and the combination of backward and forward reasoning, the proposed model can generate automatic dialogues between agents (even if, for the moment, the set of performatives involved is relatively small). The dialogue policies defined in this paper are application-independent and more general compared to the similar concept of dialogue constraints presented in the work of Sadri et al. [12] (as noted above, dialogue policies enable us specify various personalities of agents). Moreover, our definitions of performatives and dialogue contexts enable us modeling nested dialogues. Note also that our framework clearly specifies the way dialogues are generated (this feature does not appear in [9]). Third, the proposed representation of dialogue contexts allows the verification of the ability of an agent to participate in a dialogue on a specific topic (concerning the decision to enter or not in a particular dialogue type), which is also another difference with [12] and, like in [5], enables participants to be aware of the nature of the dialogue to be undertaken. Finally, integrating issues arising from the proposed knowledge structure, reasoning mechanisms and execution cycle, our work provides an operational framework for conversational agents (probably more than those appearing in [1, 3, 5, 8, 9]).

Our primary future work direction concerns the automation of moves from one dialogue type to another. Moreover, inspired by [2, 8], we plan to integrate agents mental attitudes (beliefs, desires, intentions) in our framework. Another direction concerns the definition of more properties for our framework and its enrichment with more performatives and dialogue policies (including policies for nested dialogues).

Acknowledgements: We would like to thank the referees for their helpful comments, which helped us improve this paper. Also, K. Karenos, L.Michael and C. Christofi for their help in the implementation of our framework.

REFERENCES

[1] Amgoud, L., Parsons, S. and Maudet, N. Arguments, dialogue and negotiation. In *Proc. of ECAI 2000*, Berlin, 2000, 338-342.

[2] Cohen, P. and Levesque, H. Communicative actions for artificial agents. In *Proc. of ICMAS'95,* San Francisco, AAAI Press, 1995.

[3] Hitchcock, D., McBurney, P., and Parsons, S. A Framework for deliberation dialogues, Argumentation and its Applications. In *Proc. of the Fourth Biennial Conference of the Ontario Society for the Study of Argumentation*, 2001.

[4] Kowalski, R.A., and Sadri, F. From logic programming to multi-agent systems. *Annals of Mathematics and AI*, 1999.

[5] McBurney, P., and Parsons, S. A formal framework for inter-agent dialogues. In *Proc. of AGENTS'01*, Montreal, 2001, ACM Press, NY.

[6] McBurney, P., and Parsons, S. Agent Ludens: Games for agent dialogues. In *Proceedings of AAAI Spring Symposium on Game Theoretic and Decision Theoretic Agents (GTDT2001)*, P. Gmytrasiewicz and S. Parsons (eds.), 2001.

[7] Parsons, S. and Jennings, N.R. Negotiation through argumentation - a preliminary report. In *Proc. of ICMAS '96*, 1996, 267-274.

[8] Pitt, J.V. and Mamdani, A. A Protocol-Based Semantics for an Agent Communication Language. In *Proceedings of IJCAI'99*, Stockholm, Morgan-Kaufmann Publishers, 1999, 486–491.

[9] Reed, C. Dialogues frames in agent communication. In *Proceedings of 3rd Intern. Conference on Multi Agent Systems*, 1998, 246-253.

[10] Russell, S. and Norvig P. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, NJ, 1995.

[11] Sabater, J., Sierra, C., Parsons, S. and Jennings, N. R. Engineering executable agents using multi-context systems. *Journal of Logic and Computation* 12, 2002.

[12] Sadri F., Toni, F. and Torroni, P. Dialogues for negotiation: agent varieties and dialogue sequences. In *Proc. of ATAL-2001*, 2001.

[13] Sierra C., Jennings, N.R., Noriega, P., and Parsons, S. A Framework for argumentation-based negotiation. In *ATAL '97*, 1997, 167-182.

[14] Walton, D.N. and Krabbe, E.C.W. *Commitment in dialogue: Basic Concepts of Interpersonal Reasoning*. State University of New York Press, NY, 1995