

INTER-AGENT DIALOGUES IN ELECTRONIC MARKETPLACES

NIKOS KARACAPILIDIS

MEAD, IMIS Lab, University of Patras, Greece

PAVLOS MORAÏTIS

Department of Computer Science, University of Cyprus, Cyprus

This paper presents an overall framework for carrying out different types of dialogues between intelligent and autonomous agents acting in an electronic marketplace. Such dialogues take place during various commercial transactions concerning requests and offers of products and services. The proposed dialogue framework has been adopted in the communication and collaboration protocols of an already implemented system, which enables buyers and sellers delegate a variety of tasks to their personal agents. Much attention has been paid to the personalization of collaborative agents, which may permanently live and interact in the market representing their owners' interests. Our overall approach builds on a modular decomposition of the agents involved, and a formal and operational modeling of the associated dialogues. Features of our framework are demonstrated through an illustrative example of dialogues deployed during interagent transactions on the establishment of a combined reservation for dinner and a movie. The main contribution of this work is that the proposed framework is capable to represent disparate dialogues taking place among agents having adopted diverse strategies for carrying out e-commerce transactions.

Key words: interagent dialogues, agent-based marketplace, electronic commerce.

1. INTRODUCTION

Intelligent agents technology plays a decisive role in various electronic business applications (Guttman, Moukas, and Meas 1998; Nwana et al. 1998). Such agents can efficiently support tasks incorporated in a series of related business models such as e-auctions, e-malls, information brokers, and e-marketplaces (Timmers 1998). Their ability to provide functionalities in an autonomous, proactive, social, and adaptive fashion offers greater flexibility than traditional components when developing such applications. Research work on the use of intelligent agents in e-business applications basically focuses on making agents "smarter," aiming at providing a wider spectrum of functionalities and developing frameworks, i.e., marketplaces, in which agents can efficiently operate.

This paper exploits a formal and operational conversation model, first appeared in Karacapilidis and Moraïtis (2002), and describes its application in commercial transactions concerning buying and selling of services in an agent-based electronic market system. The proposed model constitutes the basis for the establishment of the communication and collaboration protocols of an e-market system and has been applied to an extension of an already implemented system¹ to represent interactions taking place between artificial agents of the system. Inspired by the typology introduced by Walton and Krabbe (1995), we associate commercial transactions with the messages passed between artificial agents, and describe a conversation protocol that is able to support buying and selling services. In the proposed system, human users dedicate to their artificial agents, which "live" permanently in the e-market, the task to communicate with other artificial agents representing providers of services (e.g., reservation of cinema tickets or tables at a restaurant) to buy services. All artificial agents use a common conversation protocol that is rich enough to take into account a large spectrum of communicative acts necessary for the dialogue a buyer and a seller agent must have to conclude the selling of a service that completely satisfies the preferences of the buyer.

Address correspondence to Nikos Karacapilidis at MEAD, IMIS Lab, University of Patras, 26504 Rion Patras, Greece; e-mail: nikos@mech.upatras.gr

¹For a detailed description of the system see Karacapilidis and Moraïtis (2001a,b).

Compared to other agent communication protocols, 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 and nested dialogues, which are often necessary in an electronic marketplace environment, in a unified way. Moreover, the related concepts of *dialogue context* and *dialogue policies* allow us implement various cooperation protocols and profiles of trading agents.

The paper is organized as follows: Section 2 gives an overview of our previous work on the implementation of an agent-based e-market system. Section 3 reports on the modular decomposition of our agents focusing on the structure of their knowledge base. Section 4 then discusses the modeling of interagent dialogues by defining the concepts of communication performatives, dialogue contexts, and dialogue policies, which are actually the rules serving the automated generation of the appropriate illocutionary acts. Section 5 provides an illustrative example that demonstrates important features of our framework. Finally, Section 6 comments on related work and outlines future work directions.

2. THE AGENT-BASED MARKETPLACE

The software agents in our electronic marketplace are classified into two broad categories, namely *buyers* and *sellers*, while each of them assists an actor (*customer* or *merchant*, respectively). Using the system, actors can delegate a series of tasks to their personal intelligent agents, which play a role of *artificial employees*. Moreover, there are two broad categories of messages passed; the first one concerns interactions between an actor and his/her agent, while the other concerns interactions taking place between agents. Orthogonal to that, dialogues taking place in our framework may be classified as either concerning the personalization of agents or a buying/selling transaction. In order for the paper to be self-contained, this section provides an overview of our agent-based electronic marketplace and briefly presents all sorts of dialogues taking place. It should be made clear though that the rest of this paper deals only with dialogues concerning buying/selling transactions for services, whereas the associated interactions take place only between artificial agents (buyers and sellers called afterward providers).

Contrary to the majority of the agent-mediated artificial market systems, our overall approach is not based on preclassified ads (Karacapilidis and Moraïtis 2001b); instead, the systems' agents collaborate in real-time mode. More specifically, the overall approach builds on the feature of proactiveness and semiautonomy of all artificial agents involved. Upon the current setting, agents can take the initiative to contact their actors to start a transaction that seems "interesting" to them (e.g., when a new product or service that matches one's profile appears in the market), or trigger an actor's action (e.g., they can inform their merchant that a specific offer is of no interest in the market for the last month). Semiautonomy of agents has been chosen to assure the right level of control for the actions they could take (we argue that a fully autonomous agent could cause problems in a commercial setting). Second, the approach followed is based on a long (or even permanent) existence of agents in the e-market. In other words, agents do not "live" only during a specific transaction but much longer, depending upon the subscription paid by their owners at the time they were launched (i.e., an actor may "hire" an agent for a month, a year, etc.). This is highly associated with the personalization of the agents involved through the maintenance of each actor's comprehensive *profile*. Third, the system enables the e-market's seller agents to refine (some of) a customer's purchase criteria during a transaction, argue in favor or against them, or even bring up new information to persuade him/her to accept their offers. Finally, the system is able to handle incomplete, inconsistent, and conflicting information during a purchase transaction, and

perform a progressive synthesis and comparative evaluation (across a set of attributes) of the existing proposals.

2.1. Buying and Selling of Products

Transactions taking place in our system may be initiated either by a human (customer or merchant) or an artificial agent. To further analyze the first case, consider a customer looking for a certain good; he/she is able to contact his/her purchaser agent and initiate a purchase transaction. In turn, the purchaser agent requests offers (from all or some seller agents) that may fulfill its actor's interests. Whenever a match between a purchaser and a seller agent is established, the latter gets information about the customer's buying criteria, preferences that may hold among them, as well as constraints explicitly imposed. By getting such a request, and presuming that the appropriate information exists in its knowledge base (selling database), a seller agent can directly build and propose an offer that is as close as possible to the purchase request. Otherwise (i.e., not enough information in the database), it has first to contact its merchant for an update of the related specifications.

Having collected a set of such offers, a purchaser agent has to consider and evaluate them all, the aim being to eventually recommend the best one to its user. At this stage, messages can be exchanged between the purchaser agent and the customer in order for the agent to get supplementary information regarding the customer's opinion about features, preferences and arguments introduced by a seller agent. The above exchange of messages takes place through user-friendly interfaces (web pages). Having been informed about the outcome of the related interagent dialogue, a customer can finally make a decision about the acceptance or rejection of a merchant's proposal, which is forwarded to all interested parties.

The point of view of each seller agent is expected to be different, representing the interests of his owner. In many cases, even conflicts among them are inevitable. For instance, during a dialogue about purchasing a car, a certain seller agent (acting for a provider who sells sport cars) may argue in favor of speed paying less attention to features concerning price and safety. On the contrary, a second seller agent (acting for a provider who sells family cars) may argue against speed, while promoting features such as comfort and reliability. In other words, before responding to a purchaser agent's request, each seller agent would have tailored its offer according to the range of products or services at hand. Moreover, each seller agent may adopt its own *strategy* and, subsequently, propose an offer that fulfils (some of) the purchaser agent's goals at a certain level. Offers may also differ about the relative values of criteria. In addition, the purchaser and the seller agents may have arguments supporting or against alternative solutions. Finally, before making a decision, the purchaser agent may have to confront the existence of insufficient information; that is, information that would be useful for making a decision is missing (for instance, consider the case where no price has been provided by a seller agent, while this criterion is important for a purchaser agent to make a decision).

As discussed above, transactions in our electronic marketplace may be also initiated by artificial agents. To give an example, consider a merchant asking his/her seller agent to broadcast (or selectively send) an offer for a certain product or service. Due to the capabilities of our system's intelligent agents to be proactive and semi-autonomous, a purchaser agent whose profile matches to a merchant's offer may take the initiative to contact its actor and ask his/her opinion to go ahead (either to purchase directly the certain product or service, or retrieve related offers and evaluate them all together). Similarly acting, a seller agent, when noticing that the market's purchaser agents continuously discard its offers due to their prices, can suggest its actor to lower them. Similarly, our framework supports dialogues concerning

transactions taking place during the introduction of a new product or service, as well as the advent of a new agent in the marketplace.

2.2. Buying and Selling of Services

Having presented the general structure of our e-marketplace, this section focuses on the case of buying and selling of services, which will be used to describe the applicability of the conversation protocol presented by Karacapilidis and Moraitis (2002). In our system, a (human) user wanting to find a service (for instance, to reserve a table at a restaurant and/or some movie tickets) interacts with his/her artificial agent to describe the service he/she is looking for. Using his “acquaintances” in the e-market (i.e., service providers), the artificial agent looks for the most appropriate one to first, establish a dialogue with him, and then submit a query with the characteristics of the service required. If the provider is able to fully satisfy the query submitted (i.e., all criteria and features can be satisfied), he informs the service demander about the solution found and the dialogue is over. Otherwise, that is only a part of the service requested can be satisfied, a subsidiary dialogue may start between the demander and provider agents aiming at finding alternatives that satisfy the request (such alternatives reside in the knowledge base of the providers). If they arrive to an agreement, the dialogue is over; otherwise, the transaction fails and the dialogue is closed. As in the case of buying/selling goods, artificial agents can also interact with their (human) users to get further information about the alternatives suggested each time. In this paper, we present transactions carried out between artificial agents only; more specifically, we describe the way they autonomously convey dialogues about alternatives, and we associate it with the representation of their profiles and behavior.

It should be noted here that the related list of “acquaintances” is automatically provided by the marketplace platform (which launches the software agents). More specifically, each time a new seller (buyer) agent is launched, the marketplace platform provides him with the addresses of the already launched buyer (seller) agents. The new agents then send special messages to their “acquaintances” to inform them about their presence, which in turn allows the latter to update their own lists. The above approach eliminates the unnecessary reception of messages from agents of the same type, which may happen upon the adoption of other protocols not distinguishing among different types of agents (e.g., the Contract Net Protocol (Smith 1980)).

3. STRUCTURE OF AGENTS

To clearly describe the concept of dialogue frames, we describe in this section the structure of agents in our framework. The basic assumption we have made is that an agent \mathbf{Ag} is composed of a set of modules Δ_x that implement its overall behavior. More specifically, each module δ_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 modules of Δ_x . Agents interact with their environment through their respective communication modules.

Finally, to serve its role, we assume that each module δ_x (i.e., apart from the communication module) is equipped with a knowledge base $\mathbf{K}(\delta_x)$. 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.

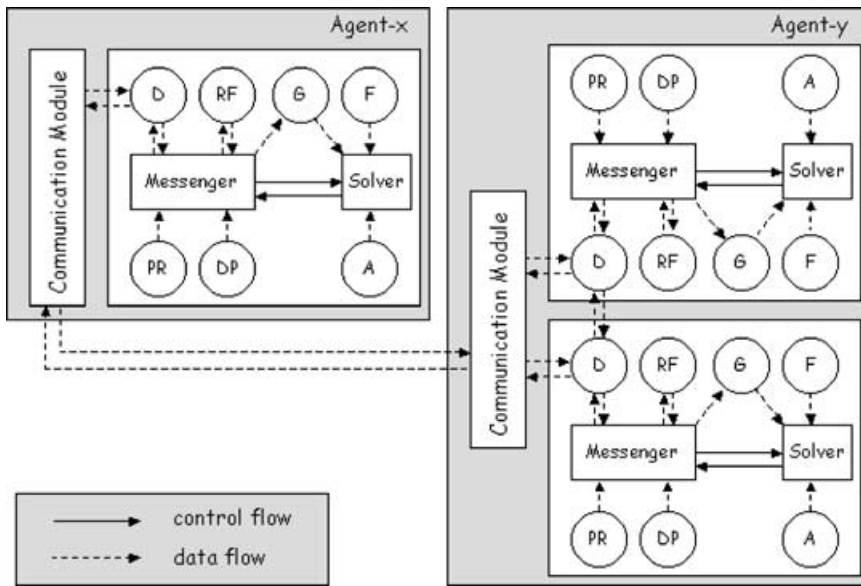


FIGURE 1. Structure of agents.

Definition 1. A knowledge base is a tuple $\langle F, G, A, \text{solver}, DP, PR, \text{messenger}, RF, D \rangle$, where (see Figure 1):

- F contains *application-specific knowledge (facts)* related to the role of the module (e.g., for a deliberation module, such 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 is placed in 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 Section 4). These policies, which are used by the *messenger*, actually regulate the dialogues, in that they explicitly specify the response to be sent by an agent after he has received a specific message.
- 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 and/or in case of embedded dialogues, between two modules of the same agent). It is implemented as a queue that is emptied after the end of each dialogue.

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.

4. MODELING DIALOGUES

In the system, briefly described in Section 2, artificial agents collaborate with either their owners (i.e., human users) or their peers (which also represent the interests of their owners). Agents interact through disparate types of dialogues based on the exchange of messages of various types. Dialogue types actually specify which module of an agent will be used upon the reception of a certain message from another agent. An agent may either accept or reject a request to get involved in a particular dialogue type. The set of dialogue types (and apparently the set of different modules an agent consists of) is set by the user, who also has to set up the appropriate knowledge base, and can be expanded at his/her wish. The current stage of our implementation is based on the well-trying and broadly accepted dialogue types introduced by Walton and Krabbe (1995).

In the current implementation stage, our framework addresses only pairwise dialogues. Moreover, an agent's module may be involved in just one dialogue at each time. A dialogue is always initiated by the exchange of specific messages that install the appropriate context of the transaction under consideration. Each time an agent receives a message, it immediately knows what module he must activate in order to set up the most appropriate dialogue. This is achieved through the explicit definition of the type of the dialogue, in that each such type is explicitly associated with an agent's module (see Definition 4).

Dialogue between agents can be a complex and possibly a nested process. Our interagent dialogue mechanism allows agents to shift seamlessly between the various dialogue types. Exploiting the conversation protocol presented in Karacapilidis and Moraitis (2002) for the needs of the overall framework presented in this paper, we formally define below the concepts of *message* and *dialogue policy*.

Definition 2. A *message* is an instance of a schema of the form $\text{Msg}=(id, P)$, 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 $\{\text{propose, accept, request, assert, refuse, challenge, reject}\}$. 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.² As shown below (see Definition 3), these acts will be exploited by the agent's mechanisms to serve the evolution of a dialogue.
- 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 $\sigma = \langle \text{sentence} [\text{support}] \rangle$ 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 and are modeled to keep track of the evolution of a dialogue.

²A detailed presentation and discussion about the semantics of illocutionary acts can be found in Searle (1969) and Searle and Vanderveken (1985).

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 $\sigma(\sigma')$ is the subject of the performative, as described above.
- C , hereafter referred to as *condition*.

As mentioned above, a dialogue policy is a part of the knowledge base of each module of an agent. Such policies actually define the personality (behavior) of each agent, in that they specify the attitude they demonstrate upon the reception of a certain performative (message). Consequently, in the marketplace environment described in this paper, such policies are able to characterize the profile of each trading agent. Moreover, being involved in the definition of what is the next message to be sent by an agent y , after the reception of a specific message coming from another agent x , a dialogue policy serves the automation of a dialogue.

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):

$$SoP = \{ \text{request} \rightarrow \text{assert}, \text{request} \rightarrow \text{refuse}, \text{propose} \rightarrow \text{accept}, \text{propose} \rightarrow \text{refuse}, \text{propose} \rightarrow \text{propose}, \text{refuse} \rightarrow \text{challenge}, \text{challenge} \rightarrow \text{assert}, \text{assert} \rightarrow \text{accept}, \text{assert} \rightarrow \text{reject} \}.$$

It should be noted here that the above set also corresponds to the current stage of our implementation; it can be easily expanded (like what holds for the illocutionary acts) to serve more elaborated cases. The dialogue policies corresponding to a usual agent's trading profile are formally presented below (condition C appears within the square brackets):

$$DP1: \text{request}(x, y, \sigma, T) \wedge [K(\delta_y) \vdash G^\sigma] \Rightarrow \text{assert}(y, x, \sigma, T+1)$$

$$DP2: \text{request}(x, y, \sigma, T) \wedge [K(\delta_y) \not\vdash G^\sigma] \Rightarrow \text{refuse}(y, x, \sigma, T+1)$$

The first two dialogue policies are rather straightforward. Upon acceptance of a **request** message from an agent x , agent y checks whether G^σ can be entailed by his knowledge base (in fact, the subject σ of a message becomes a goal G^σ). If yes, he sends back an **assert** message; otherwise, he sends a **refuse** one:

$$DP3: \text{propose}(x, y, \sigma, T) \wedge [K(\delta_y) \vdash G^\sigma] \Rightarrow \text{accept}(y, x, \sigma, T+1)$$

Like wise, upon acceptance of a **propose** message from an agent x , agent y sends back an **accept** message provided that G^σ can be entailed by his knowledge base:

$$DP4: \text{propose}(x, y, \sigma, T) \wedge [K(\delta_y) \not\vdash G^\sigma \wedge \{ \exists \sigma' \in K(\delta_y) \mid (\sigma' \succ^{Pr} \sigma) \wedge (K(\delta_y) \vdash G^{\sigma'}) \}] \Rightarrow \text{propose}(y, x, \sigma', T+1)$$

The condition imposed in DP4 means that if $K(\delta_y)$ does not entail G^σ (i.e., the goal associated to the subject σ of the received performative), it is checked whether there exist both another σ' belonging to $K(\delta_y)$ and a preference in PR stating that σ' is preferable than σ , such that

$\mathcal{G}^{\sigma'}$ can be entailed by $K(\delta_y)$. In case that there are more than one σ' 's that satisfy the above, all of them will be proposed:

$$\text{DP5: propose } (x, y, \sigma, T) \wedge [K(\delta_y) \vdash \mathcal{G}^{\sigma} \wedge \{\nexists \sigma' \in K(\delta_y) \mid (\sigma' \succ^{\text{pr}} \sigma) \wedge (K(\delta_y) \vdash \mathcal{G}^{\sigma'})\}] \\ \Rightarrow \text{refuse } (y, x, \sigma, T+1)$$

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

$$\text{DP6: refuse } (x, y, \sigma, T) \wedge [\text{support } (\sigma) = \emptyset] \Rightarrow \text{challenge } (y, x, \sigma, T+1)$$

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):

$$\text{DP7: challenge } (x, y, \sigma, T) \wedge [\exists \text{reason} \in \text{RF}(y) \mid \text{reason} \vdash (\neg\sigma)] \\ \Rightarrow \text{assert } (y, x, \sigma, T+1)$$

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. It should be made clear here that the above procedure is performed by the recipient of the **challenge** message:

$$\text{DP8: assert } (x, y, \sigma, T) \wedge [\text{support } (\sigma) \neq \emptyset \wedge \{\nexists \text{support}' \in K(\delta_y) \mid \\ \text{support}' \vdash (\neg\text{support})\}] \rightarrow \text{accept}(y, x, \sigma, T+1)$$

$$\text{DP9: assert } (x, y, \sigma, T) \wedge [\text{support } (\sigma) \neq \emptyset \wedge \{\exists \text{support}' \in K(\delta_y) \mid \\ \text{support}' \vdash (\neg\text{support})\}] \rightarrow \text{reject}(y, x, \sigma, T+1)$$

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).

Definition 4. Given that agents convey knowledge in their constituent modules, as described in the previous section, and inspired by the work presented in Reed (1998), and Walton and Krabbe (1995), we define a *dialogue context* as a tuple $\text{DC}=(\dagger, (\tau, \mathbf{M}))$, where:

- \dagger is the *type* of the dialogue ($\dagger \in \{\text{deliberation, negotiation, persuasion, ...}\}$);
- τ is the *topic* of the dialogue (i.e., what agents discuss about)³;
- \mathbf{M} is the *medium* used for the dialogue, which may refer to messages exchanged between two conversational agents x and y directly (denoted by **Direct**), through a *mediator* z (denoted by **med(z)**), or shared via a common memory w (such as a *blackboard*).⁴

³Dialogue topics actually serve the organization of the knowledge base of each agent's module, in that facts and actions residing in a knowledge base are associated with a particular topic.

⁴In the current implementation of our system, messages between agents are exchanged directly. However, especially in e-business environments, messages from buyer and seller agents may be passed through a middle agent (e.g., broker, matchmaker), see (Klusck and Sycara 2001).

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 (appropriate reasoning mechanisms are deployed by messenger and solver; see Section 3 and the example illustrated in Section 5). 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. Moreover, agents may often 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 to negotiate with his peer about the list of criteria to be considered in their decision (such “moves” from one discussion type to another are triggered by actions residing in an agent’s knowledge base). In this case, the initial dialogue context would be $DC_k=(deliberate, (car_purchase, de(x, y)))$, where a new context $DC_m=(negotiate, (criteria, de(x, y)))$ 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 (1995), assuming that dialogue types may be nested.

More specifically, it is the type \dagger of a dialogue frame that triggers the appropriate module, provided that the topic τ belongs to the beliefs B of the knowledge base $K(\delta_x)$, where δ_x is the module of agent x that corresponds to the dialogue type \dagger . The rationale of the above is that an agent must have the appropriate module and be equipped with the necessary knowledge 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 with a meteorologist agent about weather forecasting).

The proposed modularity (different sets of modules may be included in two agents) together with the engagement of different dialogue types enables us to associate different roles for each agent participating in the system. Even two agents consisting of the same set of modules may expose a different attitude, provided that the way their modules interact is not the same. For instance, an agent x may always invoke its negotiation module when its persuasion module is unable to conclude an issue, while another agent y may prefer to first invoke its information seeking module, thus trying to eliminate cases where knowledge is not clearly specified or narrow the domains of some values.

Definition 5. Adapting the approach followed by Sadri, Toni, and Torroni (2001) to our framework, *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 . More formally, for all $dp \in DP$, it holds $(K(\delta_y) \cup p_i) \not\models \text{body}(dp)$.⁵ It should be noted here that this definition could be further elaborated, if one wants to also capture abnormal termination cases (concerning communication failure between agents).

5. DIALOGUES FOR BUYING/SELLING TRANSACTIONS

This section demonstrates the applicability of our framework for interagent dialogues through an illustrative example. Let agent Z act toward making some reservations for his owner, who is actually a user of our system that looks for some services. His goal is to find a solution that combines dining at a good restaurant and going to the movies. To achieve this goal, Z contacts agents X and Y , which act as representatives (information providers) of the city’s cinemas and restaurants, respectively. As noted in the previous section, to set up a dialogue, an agent has to send a message indicating the desired dialogue frame in its body. In our example, such a message is

⁵Note that a dialogue policy dp is composed of two parts, namely $\text{body}(dp)$ and $\text{head}(dp)$, while $\text{body}(dp) \Rightarrow \text{head}(dp)$.

request(Z, X, <Deliberation (Topic:Movies_Reservation, Direct)>, T)

What is actually requested by Z is to involve X in a deliberation dialogue about the “Movies_Reservation” topic (Z is aware that X provides such services). In turn, agent X confirms its ability to participate in such a dialogue by sending the message

accept(X, Z, <Deliberation (Topic:Movies_Reservation, Direct)>, T+1)

Upon receipt of this confirmation, Z is able to submit his request. To follow the example, we first need to describe the knowledge existing in the related modules of each agent involved. The sets P, F, and A below include the definitions of the predicates, facts, and actions, respectively, used by each agent in this example (i.e., P_Z stands for the set of predicates of agent Z).⁶

Agent Z

- P_Z = {P_{Z1}: \forall avail_friend, \forall n, Avail_friends(avail_friend, n);
P_{Z2}: \forall good_restaurant, Good_restaurant(good_restaurant);
P_{Z3}: \forall dinner_time, Dinner_Time(dinner_time);
P_{Z4}: \forall fav_rest_cat, Fav_rest_cat(fav_rest_cat);
P_{Z5}: \forall good_film, Good_film(good_film);
P_{Z6}: \forall film_time, Film_time(film_time);
P_{Z7}: \forall fav_film_cat, Fav_film_cat(fav_film_cat);
P_{Z8}: \forall cinema, \forall fav_c_area, C_Area(cinema, fav_c_area);
P_{Z9}: \forall restaurant, \forall fav_r_area, R_Area(restaurant, fav_r_area);
P_{Z10}: \forall place, \forall time, Being_at(Z, place, time)}
- F_Z = {F_{Z1}: Avail_friends(Jacqueline, 1);
F_{Z2}: Good_restaurant(PARIS.16);
F_{Z3}: Dinner_Time(20.00);
F_{Z4}: Fav_rest_cat(FRENCH);
F_{Z5}: Good_film(LEON);
F_{Z6}: Good_film(LA_CONFIDENTIAL);
F_{Z7}: Film_time(22.00);
F_{Z8}: Fav_film_cat(ACTION);
F_{Z9}: C_Area(cinema, QUARTIER_LATIN);
F_{Z10}: R_Area(restaurant, QUARTIER_LATIN)}
- A_Z = {A_{Z1}: \forall friend, \forall n, \forall rest, \forall rest_cat, \forall fav_r_area, \forall dinner_time,
Avail_friends(friend, n) \wedge (Equal(n, 0) \wedge Choose(rest, rest_cat,
fav_r_area, dinner_time) \Rightarrow Want_dinner(Z, rest, rest_cat, dinner_time,
fav_r_area, n+1);
A_{Z2}: \forall rest, \forall rest_cat, \forall fav_r_area, \forall dinner_time, R_Area(rest, fav_r_area)
 \wedge Dinner_time(dinner_time) \wedge Fav_rest_cat(rest_cat) \Rightarrow Choose(rest,
rest_cat, fav_r_area, dinner_time);
A_{Z3}: \forall provider, \forall dinner_time, \forall restaurant, \forall fav_r_area, \forall n,
Know(provider) \wedge Can_find(provider, restaurant, fav_r_area) \wedge
Want_dinner(Z, rest, rest_cat, dinner_time, fav_r_area, n) \Rightarrow

⁶The meaning of the abbreviations used in this example is as follows: avail_friend stands for “available friend,” rest_cat for “restaurant category,” film_cat for “film category,” fav_rest_cat for “favorite restaurant category,” fav_film_cat for “favorite film category,” fav_c_area for “favorite cinema area,” fav_r_area for “favorite restaurant area,” C_area for “cinema area,” R_area for “restaurant area,” and av_places for “available places.”

request(Z, provider, <Have_dinner(Z, rest, rest_cat, dinner_time, fav_r_area, n)>);

- A_{Z4}: $\forall \text{rest}, \forall \text{rest_cat}, \forall t, \forall q, \forall p, \text{dinner_time}=t \wedge \text{fav_r_area}=q \wedge \text{good_restaurant}=\text{rest} \wedge \text{fav_rest_cat}=\text{rest_cat} \wedge p=n+1 \Rightarrow \text{Have_dinner}(Z, \text{rest}, \text{rest_cat}, t, q, p)$;
- A_{Z5}: $\forall \text{friend}, \forall \text{film}, \forall n, \forall \text{cinema}, \forall \text{fav_c_area}, \forall \text{film_cat}, \forall \text{film_time}, \text{Avail_friends}(\text{friend}, n) \wedge \neg \text{Equal}(n, 0) \wedge \text{Choose}(\text{film}, \text{cinema}, \text{fav_c_area}, \text{film_cat}, \text{film_time}) \Rightarrow \text{Want_see_film}(Z, \text{film}, \text{cinema}, \text{film_time}, \text{film_cat}, \text{fav_c_area}, n+1)$;
- A_{Z6}: $\forall \text{cinema}, \forall \text{fav_c_area}, \forall \text{film_cat}, \forall \text{film_time}, \text{C_Area}(\text{cinema}, \text{fav_c_area}) \wedge \text{Film_time}(\text{film_time}) \wedge \text{Fav_film_cat}(\text{film_cat}) \Rightarrow \text{Choose}(\text{film}, \text{cinema}, \text{fav_c_area}, \text{film_cat}, \text{film_time})$;
- A_{Z7}: $\forall \text{provider}, \forall \text{film_time}, \forall \text{film}, \forall \text{cinema}, \forall \text{film_cat}, \forall \text{fav_c_area}, \forall n, \text{Can_find}(\text{provider}, \text{film}, \text{cinema}) \wedge \text{Know}(\text{provider}) \wedge \text{Want_see_film}(Z, \text{film}, \text{cinema}, \text{film_time}, \text{film_cat}, \text{fav_c_area}, n) \Rightarrow \text{request}(Z, \text{provider}, \text{<Go_cinema}(Z, \text{film}, \text{cinema}, \text{film_time}, \text{film_cat}, \text{fav_c_area}, n)>)$;
- A_{Z8}: $\forall \text{film}, \forall \text{cinema}, \forall t, \forall k, \forall q, \forall p, \text{film_time}=t \wedge \text{fav_c_area}=q \wedge \text{good_film}=\text{film} \wedge \text{fav_film_cat}=k \wedge p=n+1 \Rightarrow \text{Go_Cinema}(Z, \text{film}, \text{cinema}, t, k, q, p)$;
- A_{Z9}: $\forall x, \forall \text{place_1}, \forall \text{place_2}, \text{place_1} \neq \text{place_2}, \forall t_1, \forall t_2, t_1=t_2, \neg \text{Being_at}(x, \text{place_1}, t_1) \Rightarrow \text{Being_at}(x, \text{place_2}, t_2)$

Agent X

- $P_X = \{P_{X1}: \forall \text{film}, \forall \text{cinema}, \forall \text{film_cat}, \forall \text{area}, \forall \text{film_time_1}, \forall \text{av_places_1}, \forall \text{film_time_2}, \forall \text{av_places_2}, \text{Film}(\text{film}, \text{cinema}, \text{film_cat}, \text{area}, \text{film_time_1}, \text{av_places_1}, \text{film_time_2}, \text{av_places_2});$
 $P_{X2}: \forall \text{cinema}, \forall \text{place}, \text{Is}(\text{cinema}, \text{place});$
 $P_{X3}: \forall \text{restaurant}, \forall \text{place}, \text{Is}(\text{restaurant}, \text{place});\}$
- $F_X = \{F_{X1}: \text{Film}(\text{LEON}, \text{ABC}, \text{ACTION}, \text{QUARTIER_LATIN}, 20_00, 5, 22_00, 20);$
 $F_{X2}: \text{Film}(\text{LA_CONFIDENTIAL}, \text{ELYSEE}, \text{ACTION}, \text{QUARTIER_LATIN}, 20_00, 5, 22_00, 0)\}$
- $A_X = \{A_{X1}: \forall \text{client}, \forall \text{film}, \forall \text{cinema}, \forall t, \forall k, \forall q, \forall p, \text{film_cat}=k \wedge \text{area}=q \wedge ((\text{film_time_1}=t \wedge \text{av_places_1}>p) \vee (\text{film_time_2}=t \wedge \text{av_places_2}>p)) \Rightarrow \text{Go_Cinema}(\text{client}, \text{film}, \text{cinema}, t, k, q, p);$
 $A_{X2}: \forall \text{client}, \forall t, \forall p, \forall k, \forall q, \text{film_cat}=k \wedge \text{area}=q \wedge ((\text{film_time_1}=t \wedge \text{av_places_1} < p) \vee (\text{film_time_2}=t \wedge \text{av_places_2} < p)) \Rightarrow \text{request}(X, \text{client}, \text{<Negotiation}(\text{Topic}: \text{Film_time}, \text{Direct}>);$
 $A_{X3}: \forall p, \forall t, (\text{film_time_1}=t \wedge \text{av_places_1} < p) \wedge \text{av_places_2}>p \Rightarrow \text{propose}(Z, \text{client}, \text{<Being_at}(\text{client}, \text{cinema}, \text{film_time_2})[]>)$
 $A_{X4}: \forall p, \forall t, (\text{film_time_2}=t \wedge \text{av_places_2} < p) \wedge \text{av_places_1}>p \Rightarrow \text{propose}(Z, \text{client}, \text{<Being_at}(\text{client}, \text{cinema}, \text{film_time_1})[]>)$
 $A_{X5}: \forall x, \forall \text{place_1}, \forall \text{place_2}, \text{place_1} \neq \text{place_2}, \forall t_1, \forall t_2, t_1=t_2, \neg \text{Being_at}(x, \text{place_1}, t_1) \Rightarrow \text{Being_at}(x, \text{place_2}, t_2)\}$

Agent Y

- $P_Y = \{P_{Y1}: \forall \text{restaurant}, \forall \text{rest_cat}, \forall \text{area}, \forall \text{dinner_time_1}, \forall \text{av_places_1}, \forall \text{dinner_time_2}, \forall \text{av_places_2}, \forall \text{dinner_time_3}, \forall \text{av_places_3}, \text{Restaurant}(\text{restaurant}, \text{rest_cat}, \text{area}, \text{dinner_time_1}, \text{av_places_1}, \text{dinner_time_2}, \text{av_places_2}, \text{dinner_time_3}, \text{av_places_3});$

- $P_{Y2}: \forall \text{restaurant}, \forall \text{place}, \text{Is}(\text{restaurant}, \text{place});$
 $P_{Y3}: \forall \text{cinema}, \forall \text{place}, \text{Is}(\text{cinema}, \text{place})$
 $P_{Y3}: \forall \text{client}, \forall \text{restaurant}, \forall \text{dt}_1, \forall \text{avpl}_1, \forall \text{dt}_2, \forall \text{avpl}_2, \forall \text{dt}_3,$
 $\quad \forall \text{avpl}_3, \text{Reservation}(\text{client}, \text{restaurant}, \text{dt}_1, \text{avpl}_1, \text{state}_1, \text{dt}_2,$
 $\quad \text{avpl}_2, \text{state}_2, \text{dt}_3, \text{avpl}_3, \text{state}_3)$
- $F_Y = \{F_{Y1}: \text{Restaurant}(\text{PARIS}_{16}, \text{FRENCH}, \text{QUARTIER_LATIN}, 20_{00}, 0, 21_{00},$
 $5, 22_{00}, 10)\};$
 - $A_Y = \{A_{Y1}: \forall \text{client}, \forall \text{film}, \forall \text{cinema}, \forall t, \forall k, \forall q, \forall p, \text{rest_cat}=k \wedge \text{area}=q \wedge$
 $((\text{av_places}_1 > p \wedge \text{dinner_time}_1=t) \vee (\text{av_places}_2 > p \wedge \text{dinner_time}_2=t) \vee$
 $\vee (\text{av_places}_3 > p \wedge \text{dinner_time}_3=t)) \Rightarrow \text{Have_dinner}(\text{client},$
 $\text{restaurant}, t, k, q, p);$
 $A_{Y2}: \forall \text{client}, \forall t, \forall p, \forall k, \forall q, \text{rest_cat}=k \wedge \text{area}=q \wedge ((\text{dinner_time}_1 = t \wedge$
 $\text{av_places}_1 < p) \vee (\text{dinner_time}_2=t \wedge \text{av_places}_2 < p) \vee$
 $(\text{dinner_time}_3=t \wedge \text{av_places}_3 < p)) \Rightarrow \text{request}(Y, \text{client},$
 $\langle \text{Negotiation}(\text{Topic: Dinner_time, Direct}) \rangle);$
 $A_{Y3}: \forall \text{client}, \forall \text{restaurant}, \forall p, \forall t, (\text{avpl}_1 < p \wedge \text{dt}_1=t) \wedge$
 $(\text{avpl}_2 > p) \wedge (\text{avpl}_3 > p) \wedge (\text{avpl}_2 > \text{avpl}_3) \wedge (\text{state}_2=P) \Rightarrow$
 $\text{propose}(Y, \text{client}, \langle \text{Being_at}(\text{client}, \text{restaurant}, \text{dt}_2) \rangle [])$
 $A_{Y4}: \forall \text{client}, \forall \text{restaurant}, \forall p, \forall t, (\text{avpl}_1 < p \wedge \text{dt}_1=t) \wedge$
 $(\text{avpl}_2 > p) \wedge (\text{avpl}_3 > p) \wedge (\text{avpl}_2 > \text{avpl}_3) \wedge (\text{state}_2=R) \Rightarrow$
 $\text{propose}(Y, \text{client}, \langle \text{Being_at}(\text{client}, \text{restaurant}, \text{dt}_3) \rangle [])$
 $A_{Y5}: \forall \text{client}, \forall \text{restaurant}, \forall p, \forall t, (\text{avpl}_1 < p \wedge \text{dt}_1=t) \wedge$
 $(\text{avpl}_2 > p) \wedge (\text{avpl}_3 > p) \wedge (\text{avpl}_2 < \text{avpl}_3) \wedge (\text{state}_3=P) \Rightarrow$
 $\text{propose}(Y, \text{client}, \langle \text{Being_at}(\text{client}, \text{restaurant}, \text{dt}_3) \rangle [])$
 $A_{Y6}: \forall \text{client}, \forall \text{restaurant}, \forall p, \forall t, (\text{avpl}_1 < p \wedge \text{dt}_1=t) \wedge$
 $(\text{avpl}_2 > p) \wedge (\text{avpl}_3 > p) \wedge (\text{avpl}_2 < \text{avpl}_3) \wedge (\text{state}_3=R) \Rightarrow$
 $\text{propose}(Y, \text{client}, \langle \text{Being_at}(\text{client}, \text{restaurant}, \text{dt}_2) \rangle [])$
 $A_{Y7}: \forall x, \forall \text{place}_1, \forall \text{place}_2, \text{place}_1 \neq \text{place}_2, \forall t1, \forall t2, t1=t2,$
 $(\text{Being_at}(x, \text{place}_1, t1) \Rightarrow \text{Being_at}(x, \text{place}_2, t2))$

Assume now that Z sends the message $\text{request}(Z, X, \langle \text{Go_cinema}(Z, \text{film}, \text{cinema}, 22_{00}, \text{ACTION}, \text{QUARTIER_LATIN}, 2) \rangle, T+2)$, which includes the agent's preferences about the show time, film category, desired theater's area, and seats to be reserved. Having considered his knowledge base in order to satisfy the needs of Z, agent X is able to send back only one proposal (only F_{X1} can fully satisfy the request of Z; according to F_{X2} there are no seats available for the time requested), which will be conveyed by the following message:

$\text{assert}(X, Z, \langle \text{Go_cinema}(Z, \text{LEON}, \text{ABC}, 22_{00}, \text{ACTION},$
 $\text{QUARTIER_LATIN}, 2) [] \rangle, T+3)$

Due to the facts $F_{Z1}, F_{Z5}, F_{Z7}, F_{Z8}, F_{Z9}$, and action A_{Z8} agent Z decides to accept the above proposal. At the same time, Z updates his/her knowledge base, including in it the item $F_{Z11}: \text{Being_at}(Z, \text{ABC}, 22_{00})$ (this action is performed by the reasoning mechanism of the agent; its formal description is out of the scope of this paper). The following message is then sent back to agent X:

$\text{accept}(Z, X, \langle \text{Go_cinema}(Z, \text{LEON}, \text{ABC}, 22_{00}, \text{ACTION},$
 $\text{QUARTIER_LATIN}, 2) [] \rangle, T+4)$

Similarly to what happened above between agents *Z* and *X*, the message *request* (*Z*, *Y*, $\langle \text{Deliberation}(\text{Topic:Restaurant_Reservation, Direct}) \rangle$, $T+5$) will trigger agent *Y* to participate in a dialogue concerning reservation of a table at a restaurant. Agent *Y* accepts to get involved in such a dialogue by sending the message *accept* (*Y*, *Z*, $\langle \text{Deliberation}(\text{Topic:Restaurant_Reservation, Direct}) \rangle$, $T+6$). Let then agent *Z* send the message:

request(*Z*, *Y*, $\langle \text{Have_dinner}(\text{Z, restaurant, FRENCH, 20_00, QUARTIER_LATIN, 2}) [] \rangle$, $T+7$)

As above, this message conveys the agent's preferences about the restaurant category, dinner time, restaurant's area, and places to be reserved. As shown in F_{Y1} , agent *Y* can only partially satisfy the above request (i.e., regarding the restaurant category and area preferred), because no places are available for the time requested. However, *Y* knows that there are 5 and 10 places available for 21:00 and 22:00, respectively. In such cases, *Y* constructs an instance of the predicate P_{Y3} , grounded by the agent he discusses with (*client*), the restaurant found (*PARIS.16*) and tuples corresponding to the existing alternatives, i.e., pairs of dinner time and places available. Note that each such pair is accompanied by a *state* whose allowed values are *I* (impossible), *P* (possible), and *R* (refused). The first value is automatically given when there are no more available places for the associated dinner time ($avpl=0$); in case that the associated pair has been previously refused (by the agent that discusses with *Y*), the last value is given; in any other case, the *state* takes the second value. In such a way, the agent can actually decide about which alternative to propose next (see below). According to the above, a new item (fact) $F_{Y2}:\text{Reservation}(\text{Z, PARIS_16, 20_00, 0, I, 21_00, 5, P, 22_00, 10, P})$ is now (automatically) inserted in the knowledge base of *Y*. The inclusion of A_{Y2} in the knowledge base of agent *Y* demonstrates another interesting feature of our framework; having no other way to satisfy the initial request (while knowing that there are other alternatives possible), *Y* will ask *Z* to negotiate about the dinner time, by initiating a subsidiary (nested) dialogue of a different type (the first dialogue was a deliberation one):⁷

request(*Y*, *Z*, $\langle \text{Negotiation}(\text{Topic:Dinner_Time, Direct}) \rangle$, $T+8$)

Agent *Z* confirms its ability to participate in such a dialogue by sending the message:

accept(*Z*, *Y*, $\langle \text{Negotiation}(\text{Topic:Dinner_Time, Direct}) \rangle$, $T+9$)

The actions $A_{Y3}-A_{Y6}$ are then used in order for *Y* to propose an alternative solution. Actually, these are only a part of the agent theory (e.g., rules) used by its reasoning mechanism (we have included the parts corresponding to this example). The rationality of these actions is to send back an alternative that corresponds to the dinner time for which the restaurant has the maximum number of available seats (in that the agent tries to avoid peaks). Making use of A_{Y5} , agent *Y* sends to *Z* the following proposal:

propose(*Y*, *Z*, $\langle \text{Being_at}(\text{Z, PARIS_16, 22_00}) [] \rangle$, $T+10$)

As declared in the knowledge base of *Z*, the restaurant suggested is considered as a good one (see F_{Z2}), while the time proposed contradicts with the agent's knowledge about where he has already reserved to be (see F_{Z11}). Thus, agent *Z* will refuse the above proposal (see $DP5$):

refuse(*Z*, *Y*, $\langle \text{Being_at}(\text{Z, PARIS_16, 22_00}) [] \rangle$, $T+11$)

⁷Note that all other features of the request are satisfied in the knowledge base of *Y*.

Agent *Y* then challenges the above message (see DP6), attempting to get the reason led *Z* to send it:

challenge(*Y*, *Z*, <Being_at(*Z*, PARIS_16, 22_00) []>, T+12)

Agent *Z* sends back the reason of his previous act (see DP7):

assert(*Z*, *Y*, <Being_at(*Z*, PARIS_16, 22_00) [Being_at(*Z*, ABC, 22_00)]>, T+13)

With the above message, agent *Z* actually informs agent *Y* that he could not be at the restaurant PARIS_16 at 22_00, because at exactly the same time he will be at the cinema ABC. Agent *Y* cannot but accept the above assertion (see DP8), because he has no contradictory reason to the one sent by *Z* (due to A_{Y7} , P_{Y2} , P_{Y3} he knows that one cannot be at two different places at the same time). He also updates F_{Y2} (because its proposal has been rationally refused), which now takes the form: Reservation(*Z*, PARIS_16, 20_00, 0, I, 21_00, 5, P, 22_00, 10, R). Then, exploiting A_{Y6} , he proposes the last alternative he has in hand:

accept(*Y*, *Z*, <Being_at(*Z*, PARIS_16, 22_00) [Being_at(*Z*, ABC, 22_00)]>, T+14)

propose(*Y*, *Z*, <Being_at(*Z*, PARIS_16, 21_00) []>, T+15)

This time, agent *Z* accepts the alternative proposed:

accept(*Z*, *Y*, <Being_at(*Z*, PARIS_16, 21_00) []>, T+16)

Having successfully terminated the negotiation about dinner time (<Negotiation(Topic: Dinner_Time, Direct)>), agent *Y* is able now to send a message conveying the complete alternative proposed (see DP1). Finally, according to his knowledge base, agent *Z* cannot but accept it (see DP8); in such a way, the deliberation process about the reservation of a restaurant is also concluded (<Deliberation(Topic:Restaurant_Reservation, Direct)>):

assert(*Y*, *Z*, <Have_dinner(*Z*, PARIS_16, FRENCH, 21_00,
QUARTIER_LATIN, 2) []>, T+17)

accept(*Z*, *Y*, <Have_dinner(*Z*, PARIS_16, FRENCH, 21_00,
QUARTIER_LATIN, 2) []>, T+18)

Obviously, the knowledge bases of both agents have to be updated again. That is, agent *Z* updates his knowledge base inserting a new item F_{Z12} : Being_at(*Z*, PARIS_16, 21_00), while *Y* updates the item concerning the places available at the restaurant proposed (F_{Y1}).

The above example demonstrates most of the abilities of our framework. More specifically, we have shown how agents take the initiative to start a discussion and act autonomously to conclude it. The outcomes of such dialogues are forwarded to the human user (the one who wanted to use the marketplace to organize a soirée), which will make the final decision. The parts of the agents' knowledge bases included above are the ones needed to follow the example. In a real setting, the cardinality of the sets of facts of the provider agents (*X* and *Y*) would be obviously much bigger, because these would represent all existing alternatives about cinemas and restaurants (all records from the related databases used).

We should also note that in the above example, the conflicts arisen due to time aspects, which are taken into account by the theories of agents, only concern time points.

However, to consider any kind of conflict that is associated with the representation of time, one can use different temporal reasoning approaches, based—for example—on Allen’s interval algebra (Allen 1983) or on event calculus (Kowalski and Sergot 1986). Temporal reasoning in the context of our application can be also captured by adopting the disjunctive linear relations (DLR) formalism (Jonsson and Backstrom 1998). Although reasoning in the full DLR language is known to be NP-complete, we expect that in most of the domains in which our framework will be applied, temporal reasoning can be modeled in one of the tractable fragments of the DLR language, like Horn DLR.

6. CONCLUSION

We have exploited a formal and operational communication protocol for interagent dialogues, first presented in Karacapilidis and Moraitis (2002), to efficiently handle transactions taking place in an electronic marketplace. The proposed protocol has been inspired by various works in the agent technology literature (Cohen and Levesque 1995; Pitt and Mamdani 1999; Amgoud, Parsons, and Maudet 2000; Hitchcock, McBurney, and Parsons 2001; McBurney and Parsons 2001a,b). Through an illustrative example, concerning a real application about the organization of a soirée, it has been shown that the proposed conversation protocol can be used in electronic commerce applications. This is firstly due to the expressiveness of the model, which allows us to represent disparate and nested dialogues (e.g., deliberation about going out in the beginning of the dialogue and then negotiation about the dinner time, to avoid the conflict with the cinema time). Secondly, the concept of dialogue policy allow us implement various cooperation protocols and profiles of trading agents, which are ubiquitous in such environments. Even if the agents involved in the example above adopt the same set of dialogue policies, there are numerous examples (e.g., during a process of selling/buying products) where the set of dialogue policies specified for each seller agent can implement different selling strategies. Moreover, as made clear above, purchaser and seller agents may often exchange arguments in favor or against alternative solutions in an electronic marketplace. Our formalism of a communication performative takes into account this issue by providing the interacting agents with the appropriate structures to easily convey alternatives together with the associated arguments. A detailed comparison of the proposed framework with previous related work, focused on the expressiveness of the model for multiagent dialogues, can be found in Karacapilidis and Moraitis (2002).

Our artificial agents 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 automated generation and continuation of dialogues. This last issue is very important for an agent-based e-marketplace in order for it to operate with a minimum intervention of human users. In the future, we plan to use argumentation-based reasoning, by exploiting the work presented in Kakas and Moraitis (2002), to handle cases where nonmonotonic reasoning is necessary. For instance, such an approach will enable us implement negotiation dialogues supporting more complex strategies (e.g., negotiation about a product price between a buyer and a seller agent), which are also necessary in an e-commerce environment.

ACKNOWLEDGMENTS

The authors thank Bruce Spencer and three anonymous referees for their helpful comments and suggestions toward improving this paper.

REFERENCES

- ALLEN, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, **26**(1):832–843.
- AMGOUD, L., S. PARSONS, and N. MAUDET. 2000. Arguments, dialogue And negotiation. *In Proceedings of ECAI 2000 Conference, Berlin, Germany. Edited by W. Horn. IOS Press, Amsterdam, pp. 338–342.*
- COHEN, P., and H. LEVESQUE. 1995. Communicative actions for artificial agents. *In Proceedings of ICMAS 1995, San Francisco. AAAI Press.*
- GUTTMAN, R., A. MOUKAS, and P. MAES. 1998. Agent-mediated electronic commerce: A survey. *Knowledge Engineering Review*, **13**(2):143–152.
- HITCHCOCK, D., P. MCBURNEY, and S. A. PARSONS. 2001. A framework for deliberation dialogues: argumentation and its applications. *In H. V. Hansen, C. W. Tindale, J. A. Blair, and R. H. Johnson, editors. Proceedings of the Fourth Biennial Conference of the Ontario Society for Study of Argumentation (OSSA 2001). Windsor, Ontario, Canada.*
- JONSSON, P., and C. BACKSTROM. 1998. A unifying approach to temporal Constraint reasoning. *Artificial Intelligence*, **102**(1):143–155.
- KAKAS, A., and P. MORAÏTIS. 2003. Argumentation based decision making for autonomous agents. *In Proceedings of Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'03), Melbourne, Australia.*
- KARACAPILIDIS, N., and P. MORAÏTIS. 2001a. Building an agent-mediated electronic commerce system with decision analysis features. *Decision Support Systems*, **32**(1):53–69.
- KARACAPILIDIS, N., and P. MORAÏTIS. 2001b. Intelligent agents for an artificial market system. *In Proceedings of the Fifth International Conference on Autonomous Agents (Agents-2001), Montreal, Canada, May 28–June 1. ACM Press, New York, pp. 592–599.*
- KARACAPILIDIS, N., and P. MORAÏTIS. 2002. Engineering issues in inter-agent dialogues. *In Proceedings of the 15th European Conference on Artificial Intelligence (ECAI 2002), Lyon, France, July 21–26. Edited by F. van Harmelen. IOS Press, Amsterdam, pp. 58–62.*
- KLUSCH, M., and K. SYCARA. 2001. Brokering and matchmaking for coordination of agents societies: A survey. *In Coordination of Internet Agents: Models, Technologies, and Applications. Edited by A. Omicini et al. Springer-Verlag, Berlin, pp. 197–224.*
- KOWALSKI, R. A., and M. J. SERGOT. 1986. A logic-based calculus of events. *New Generation Computing*, **4**:67–95.
- MCBURNEY, P., and S. PARSONS. 2001a. A formal framework for inter-agent dialogues. *In J. P. Muller, E. Andre, S. Sen and C. Frasson (Editors): Proceedings of the Fifth International Conference on Autonomous Agents (Agents-2001), Montreal, Canada. ACM Press, New York, pp. 178–179.*
- MCBURNEY, P., and S. PARSONS. 2001b. Agent ludens: Games for agent dialogues. *In S. Parsons and P. Gmytrasiewicz (Editors): Game-Theoretic and Decision-Theoretic Agents (GTDT 2001): Proceedings of the 2001 AAAI Spring Symposium, pp. 70–77. AAAI Press, Menlo Park, CA.*
- NWANA, H., ROSENSCHEIN, J., SANDHOLM, T., SIERRA, C., MAES, P., and GUTTMANN, R. 1998. Agent-Mediated Electronic Commerce: Issues, Challenges and some Viewpoints, *In Proceedings of the Autonomous Agents 1998 Conference. ACM Press, New York, pp. 189–196.*
- PITT, J. V., and A. MAMDANI. 1999. A protocol-based semantics for an agent communication language. *In Proceedings of the IJCAI 1999 Conference, Stockholm. Morgan-Kaufmann, San Mateo, CA, pp. 486–491.*
- REED, C. A. 1998. Dialogue frames in agent communication. *In Proceedings of the Third International Conference on Multi-Agent Systems. IEEE Press, pp. 246–253.*
- SADRI, F., F. TONI, and P. TORRONI. 2001. Dialogues for negotiation: agent varieties and dialogue sequences. *In John-Jules Meyer and Milind Tambe (Editors): Intelligent Agent VIII (revised paper from the 8th International Workshop on Agent Theories, Applications, and Languages (ATAL'01), Seattle, WA, USA, August 1–3, 2001), LNAI 2333, pp. 405–421, Springer-Verlag, Berlin, Germany.*

- SEARLE, J. R. 1969. *Speech Acts*. Cambridge University Press, Cambridge, UK.
- SEARLE, J. R., and D. VANDERVEKEN. 1985. *Foundations of Illocutionary Logic*. Cambridge University Press, Cambridge, UK.
- SMITH, R. G. 1980. The contract net protocol: high level communication and control in a distributed problem solver. *IEEE Transactions on Computers* **29**(12):1104–1113.
- TIMMERS, P. 1998. Business models for electronic markets. *Electronic Markets*, **8**(2):3–8.
- WALTON, D. N., and E. C. W. KRABBE. 1995. *Commitment in Dialogue: Basic Concepts of interpersonal Reasoning*. State University of New York Press, New York.