On Abstract Task Models and Conversation Policies

Renée Elio
Department of Computing Science
University of Alberta
Edmonton, Alberta T6G 2H1
(1-403) 492-9643
ree@cs.ualberta.ca

Afsaneh Haddadi
DaimlerChrysler AG
Alt-Moabit 96A,
10559 Berlin, Germany
(+49-30) 399 82 201
Afsaneh.Haddadi@daimlerchrysler.com

ABSTRACT

It is possible to define conversation policies, such as communication or dialogue protocols, that are based strictly on what messages and, respectively, what performatives may follow each other. While such an approach has many practical applications, such protocols support only "local coherence" in a conversation. In a mixed-initiative dialogue between two agents cooperating on some joint task, there must be a "global coherence" in both the conversation and in the task they are trying to accomplish. Recognition of agent intentions about the joint task is essential for this global coherence, but there are further mechanisms needed to ensure that both local and global coherence are jointly maintained. This paper presents a general yet practical approach to designing, managing, and engineering agents that can engage in mixed-initiative dialogues. In this approach, we promote developing abstract task models and designing conversation policies in terms of such models.

1. INTRODUCTION

If software agents are going to cooperate effectively with humans on complex tasks, then they will need to participate in mixed-initiative dialogues. In the context of joint problem solving, this means that both parties can control the focus of attention, shift to a different goal, or suggest an alternative method to achieve a current goal. Cooperation denotes a kind of interaction required when each of the agents has some, but not all, the information and abilities required to accomplish a task. It is just in this instance that cooperative activity via some exchange of knowledge is necessary. This requires specifying the semantics and pragmatics of a “conversation”—a sequence of messages—that enable two agents to bring a task to completion.

There are various approaches to human-agent dialogue within the discourse understanding community. These approaches range from a commitment to full-fledged understanding of unconstrained speech input [e.g., 1,10] to screen-based interfaces, in which an underlying discourse theory determines the set of plausible next-utterances the human might make, and presents them as menu options [13,20]. Both such approaches make an explicit commitment to representing and recognizing intentions as a crucial feature for mixed-initiative cooperation, consistent in spirit with theoretical proposals for specifying a semantics for mental attitudes based on beliefs, desires, and intentions [e.g., 6, 22]. The run-time recognition and handling of intentions seems essential for human-agent cooperation and communication. For while we may define an agent communication protocol (some scheme that defines how sub-sequences of messages are to be exchanged), we cannot assume that the human knows that protocol or would be inclined to abide by it. This makes the problem of understanding and structuring conversation for cooperation more complex.

The matter of mixed-initiative dialogues between software agents and humans has a place within a broader context of multi-agent systems work on communication and conversation. Several proposals exist for defining a semantics or a set of conversational sub-units as ways of structuring multi-message sequences between two agents [3,16,17]. There is also the idea of representing dialogues and conversations among humans or software agents as state transition diagrams which dates back at least to Winograd and Flores [24]. Under this view, dialogues are automatons or state-transition networks, just like any other program, except there is more than one actor or agent that is executing the network. Transitions have to be made mutually so that conversation can take place coherently. These networks can be viewed as specifying a communication protocol, insofar as they define a set of legal transitions between messages.

However, we believe there are two drawbacks to the adoption of a state-transition approach to conversation structure. First, we believe it is insufficiently rich for specification of the messages that can be exchanged [23] and the conversation policies that govern them. Secondly, for dialogue-intensive domains it is impractical to know all the possible paths of a conversation in advance and represent them as a protocol. Protocols can only maintain what we

call *local coherence*—some unity between very short sequences of messages. When a dialogue expands beyond 2-3 message sequences, there must be some way to ensure *global coherence* to the entire conversation, i.e., a coherence to the way in which very short message sequences are, crudely put, patched together. Global coherence is required in settings where we cannot assume that both agents share a common set of protocols (as with user communication) or in settings with certain dynamics and/or complexities, which require a flexibility in agent behavior beyond what can be anticipated and encoded in a single protocol. An example might be agents negotiating about the conditions of a multi-attributed contract, where some of the conditions require concurrent negotiation with other agents going back and forth on different terms. Even if a single protocol is devised, the protocol may be too general to be of much support for the dialogue needed.

The appeal to global coherence as a feature of a conversation is implied by Grice's [12] maxim of relation, which states that speakers aim to make their contributions relevant to the ongoing conversation. In other words, each speaker's contribution to the conversation relates to the utterances that come before and follow it, so that the whole conversation is *about something* [19]. Under our view of mixed initiative dialogues between two agents, that "something" is what we call an *abstract task model*. In brief, our position is that the definition of an abstract task model essentially defines a conversation policy for a *type* of task, by defining ways to recognize and respond to a delimited set of intentions that speakers can have about the task.

The ideas presented in this paper can be summarized as follows. To achieve global and local coherence in a mixed-initiative dialogue, our work takes something of a middle approach between full-fledged natural language understanding, on the one hand, and a menu-driven approach to the next sensible utterances (as determined, for example, by a discourse theory) on the other. We address what we are calling local coherence as many others have, namely by specifying protocols for conversational sub-units. However, the realization of these protocols includes an option for a "violation" or "unexpected message." To maintain global coherence, we assume that cooperating agents share what we call an abstract task model. An abstract task model defines (i) what the agents can talk about (the objects of discourse), (ii) what they can say about those things, and most importantly, (iii) how to progress on the task and when the task is completed. The abstract task model defines intentions as goals to achieve, a general strategy for goal-ordering, and methods for achieving those goals.

In order to ensure global coherence for a lengthy conversation, an agent should be able to recognize change of conversational context by recognizing intentions. If we rely on a protocol scheme for defining semantically valid 2-3 message exchanges, a change of context necessarily implies that some currently active protocol has been "violated", in that a received message does not correspond to any well-defined, expected next state in the protocol. Thus, an agent must be designed with (i) the notion that protocols can be violated, (ii) a mechanism for recognizing the intended new context, and (iii) the ability to evaluate the implication of the new context for the resumption of the current protocol (and its associated intention). That these notions are crucial to multi-agent cooperation has been argued from a theoretical perspective [7], but solutions implied by that work require systems that are capable of very sophisticated reasoning engines. By defining the objects of discourse and what can be done to or with them, the abstract task model delimits the set of intentions the speakers can have. The agent dialogue administration methods allow deviations from the expected or allowed responses to a message within a given protocol (which are local coherence violations), by exploiting the abstract task model's definitions of the intentions that might emerge during run time.

In the remainder of this article, we present an overview of our approach, situate the key ideas in the related literature, and outline our conclusions and current directions.

2. AN ABSTRACT TASK APPROACH

We are adopting a pragmatic perspective of intention as a commitment to goal, with a specification of when and how the goal is to be pursued and when the goal is abandoned [14,15]. Intentions move an agent to act and in multi-agent systems, they can be the impetus to engage in dialogue with a collaborating agent in order to act. Thus, intentions drive the conversation and it is these intentions that make multi-message sequences about some task globally coherent. In our approach, conversational policies are defined partially by protocols that define what message primitives are expected to follow in a given state. The remainder of the conversational policy rests on the run-time recognition of intentions as triggered by unexpected messages. These so-called protocol violations are not resolved at the level of predefined transitions among language performatives (i.e., paths in protocols) but instead through intentions at the level of the task and the status of execution of the joint task.

We argue that if two agents share an abstract task model, then their messages to each other—even when a message falls outside the scope of the currently active protocol—can be interpreted with respect to intentions about the task. This brings us to our second major theme, namely that an abstract task and its analysis define the objects of discourse for the individual performatives. For us, an abstract task type is something like "scheduling," "negotiation", "database search", or "diagnosis." Similar notions of generic tasks and task models had been developed to support domain-independent methodologies and architectures for developing knowledge-based problem-solving systems [4,5]. We think that it is reasonable to assume that two agents come to a cooperative venture knowing that their (abstract) task is one of search, negotiation, diagnosis, or whatever. Regardless of what the actual domain content and

domain ontology is, two cooperating agents must share an ontology for the abstract task they are jointly solving, and this ontology is different from the ontology for the actual domain (e.g., internal medicine, geology, financial databases). It is this "meta-level" ontology for the abstract task that defines a solution state and defines how progress towards that state can be talked about. An abstract task models can then be used (i) to define and recognize at run-time a set of intentions that lend global coherence to a conversation and (ii) to define the ontology and semantics used in the construction of the actual messages. Without such a model, we argue that two agents cannot know how to begin having a conversation about a joint task, whether progress is being made on the task, or when the conversation can be ended.

We aim to recognize (abstract) paths in the conversation by pushing violations of local coherence (i.e., the arrival of a message for which the local protocol does not define a transition) to the level of intention recognition. The abstract task model assists us in this regard by defining the objects of discourse and what can be said about them.

3. SUCCESSIVE REFINEMENT SEARCH AS AN ABSTRACT TASK

In this section, we briefly outline aspects of an application domain that led us to the perspectives described above. The problem called for a system, which we here call ATS, that would assist a human (e.g., a driver in an automobile) in a variety of tasks, which included location and retrieval of information from various databases.[1] Generally speaking, ATS must identify one or more database entities that satisfy a set of constraints specified by the user. The database entities are defined by feature-value tuples. Using a database of restaurants as an example, a particular database entry corresponds to a collection of feature-value specifications like *:name Rangoon :location Palo Alto :ethnic-type Burmese :cost moderate etc.*

The first attribute of this problem is that the database is presumed to be so large that the user cannot specify a single, simple database query (e.g., "Where is the closest gas station?") We also assume that the user may not necessarily have a definitive set of constraints in mind when starting the task. It may be more accurate therefore, to characterize the process as one of exploration. This was the motivation for viewing ATS and the user as two agents cooperating in a successive refinement search, where the goal is to identify one or more database entries that satisfy a particular set (or possibly different sets) of constraints. The following example dialogue illustrates the nature of the task, where S designates the system and U designates the user.

S1: What type of location do you want to locate?
Lumber stores? Pharmacies?

U2: I want inexpensive food in San Francisco.

S3: O.K.. There are several hundred inexpensive restaurants in San Francisco.

S5: What type of food do you want?

U6: What do you mean?

S7: I mean ethnic type such as Hungarian or French.

U8: Italian.

S9: O.K. Do you want a take out or sit-down place?

U11: What about on the Peninsula?

S12: O.K.-- we'll leave San Francisco for a moment.

S13: There are 35 Italian restaurants on the Peninsula.

S14: A particular place or city will rapidly narrow down this set most rapidly.

U15: Redwood City or Menlo Park.

S16:....

In viewing the above dialogue to be about successive refinement search (and not about restaurant advising or locating), we developed an abstract task model that defined the objects of discourse that characterized dialogues about this task. These are: (i) a *domain*: a particular domain database in which entities are defined by features and values; (ii) a *constraint*: any feature that has a particular value assigned to it; (iii) a *search space*: a set of database entities that satisfy a set of constraints; (iv) s*earch-space members*: particular entities within a particular search space.

What can be said about the objects of discourse is restricted in terms of what can be done to them (and ultimately, the intentions an agent can have about them). In our analysis these actions are limited to: (i) *loading* a database to be searched, which defines the initial search space, (ii) *contracting*, or reducing the search space by specifying additional constraints that members must satisfy, (iii) *expanding* that search space by relaxing one or more constraints, and (iv) *describing* information about a particular member of the search space, about a particular constraint, or about the search space as a set. These are traditional database operations. There may be other capabilities that are unique to each agent (e.g., an agent might also compute the most-discriminating feature for a given search space.)

From our perspective, the semantics underlying the messages being exchanged in the example dialogue are defined by the ontology of successive refinement search as an abstract task. What these objects of discourse are and what can be said about them are intimately defined with

[1] ATS is a redesign [9] of a prototype effort called APA on adaptive user interfaces, pursued by the Adaptive Systems Laboratory, DaimlerBenz Research and Technology Center, Palo Alto, California. ATS's focus is not adaptability, although nothing about the approach requires or precludes it. We rename the design here to avoid confusion with work on adaptable interfaces by that laboratory.

what actions can be taken *to advance* the task. In addition, often there are certain sub-conversations for sharing and coordinating information that are only indirectly related to advancing the task. Utterances U6 and S7 in the sample dialogue above are an example of such sub-conversations.

Another key aspect about the sample dialogue above is that either agent can take the initiative in advancing the task in a new direction. Therefore, ATS must respond effectively to these "unexpected“ messages. For example, in utterance U11, the user does not provide an answer to the question posed in utterance S10, and instead shifts the direction of the search task. Intention recognition serves to support this mixed-initiative aspect of cooperation. Intentions that an agent can have about the task (and presumably express during the conversation) are limited by the objects of discourse, what can be done with them, and therefore what can be said about them. This is crucial to having a pragmatic but somewhat flexible approach to posting and recognizing intentions, for these objects of discourse serve to circumscribe the intention set.

4. THE ABSTRACT TASK DEFINES OBJECTS OF DISCOURSE

The semantics underlying the language primitives used in our framework borrow from general speech-act theory and the semantics are based on a number of pragmatic principles discussed in Haddadi [15]. A message consists of a specific message type (which we will call a performative type)[2], specific object of discourse, and partially specific content. It has the following format:

(performative $agent-name1 $agent-name2 $object-of-discourse $content)

For brevity's sake, we have omitted many message parameters such as feature keywords [18] and others for protocol administration.

The objects of discourse are constrained for each performative type as will be outlined later. The content of a performative can be another ("inner") performative or a functional call. Table 1 presents the set of performatives defined for ATS. The outermost performatives represent the general class of an utterance. In ATS, we make use of the classes *request, query,* and *inform.*

The inner performatives given in Table 1 further specialize the class, by supplying information related to the result of the task action that has been performed, the task itself or the action the speaker intends/expects the hearer to perform. The *$agent-name1* parameter refers to the speaker, sender or generally the actor of the performative, while *$agent-name2* refers to the hearer, receiver or generally the agent that would be effected by the performative.

Outer Performative	Immediately Expected Reply
Request	
Inner Performative	
Provide	(Acknowledge) + Inform
Suggest	(Acknowledge) + Inform + Accept/Reject
Query	
Inner Performative	
Provide	Inform
Confirm	Inform + Confirm/Deny
Suggest	Inform + Accept/Reject
Inform	
Inner Performative	
Provide	
Confirm	
Deny	
Accept	
Reject	

Table 1: Performatives and their Combination

The second column of Table 1—the immediately expected reply—designates the performative that would "complete" the dialogue initiated by the performative in column 1. Put another way, the information in Table 1 defines a basic state-transition definition for sub-dialogues. Here we will not discuss the issues surrounding the recognition of these performatives from a natural language utterance, and instead focus primarily on the semantics of these primitives and their associated objects of discourse.

Request. A request performative is tightly coupled with advancing the search task. The objects of discourse associated with request are (i) a system *action* that enables a search task to begin or terminate, such as loading a particular database for searching and (ii) a *constraint*, which specifies a feature-value vector according to which database entities can be identified. Most request performatives concern constraints.

Following Haddadi [15], we view a request as having an associated level of commitment. When a request is made by ATS, the system is making a pre-commitment to how the progress on the search task might be accomplished and it prompts the user for information in order to do this. System requests thus take suggest as an inner performative. A suggestion refers to a possible task strategy and it must be accepted or rejected by the other agent, in this case, the user. By supplying the information asked for, the user is committing to this computation. When a request is made by the user, the user is simultaneously committing to a computation on the search space and delivering the information necessary to execute it. User requests thus take provide as an inner performative. The objects-of-discourse

[2] Although this term is appropriated from Austin [2], we acknowledge it is being used in a different way than he and subsequent writers [21] employed it. In Austin’s use, performatives were the sort of thing that a speaker could do in making an utterance. Here, we use the term to denote the different message types that initiate or bring about different changes within the system. Our notion of “inner performative” reflects only how a further hierarchical classification of the message class is syntactically mirrored in the message format.

that may accompany suggest and provide are (i) the *domain* (e.g., restaurants, hospitals); (ii) the *value* of one or more specified features that comprise a constraint; and (iii) the search *space*. Table 2 provides some examples of system and user requests.

(i) ReqU: Lets look for a restaurant in Mid-Peninsula.
(request U S :action (initiate :task search :domain restaurants))
(request U S :constraint (provide U S :value (fv-pairs :feature location :value Mid-peninsula)))
(ii) ReqS: How about Chinese?
(request S U :constraint (suggest S U :value (fv-pairs :feature rest-type :value Chinese)))
(iii) ReqS: What kind of price range?
(request S U :constraint (provide U S :value (fv-pairs :feature price :value ?)))
(iv) ReqU: Never mind the price / I don't care about the price
(request U S :constraint (provide U S :value(expand :space $space-id (fv-pairs :feature price :value any)))

Table 2: Example Performatives for Requests

Query. A query is not about advancing a task but about exchanging information indirectly related to advancing the task. According to our abstract model, information can be exchanged about (i) system *capabilities*, that is in what domains it can assist the user with the search task (ii) the domain *knowledge base*, which includes domain-specific information, such as the range of values on a particular feature, (iii) the *database*, which includes queries about the availability of information about particular entities in the database, and (iv) *task-information*, information relevant to the current state of the task. Queries may take either provide, suggest, or confirm as an inner performative. A confirm expresses the truth or falseness with respect to a property of some object-of-discourse and it must be confirmed or denied. When a query is sent by the user agent, the system must respond with an inform followed by an appropriate inner performative. System queries often take the form of suggestions, in which case the user response must be interpreted as providing (at least) an acceptance or rejection of the suggestion. The objects-of-discourse that may accompany the inner performatives associated with queries can be (i) the *domain* (e.g., restaurants, hospitals); (ii) the current search *space* (i.e., the set of members defined by the current set of constraints); (iii) a particular *member* in the current search space or database; and (iv) the *coverage*, that is the range of values a feature may have. These constitute the objects-of-discourse that can accompany queries. See Table 3 for examples of user and system queries.

Inform. Inform messages take on the outer and inner objects-of-discourse, and a content specification, that occur in the request or query dialogue that they complete. According to the state and context of the dialogue being carried out, an utterance can result in more than one message. For instance, the first sample utterance in Table 2 would result in the two request messages (for loading the restaurant database and starting the search task with the

(i) QueU: Can you help me with finding movies?
(query U S :capability (confirm S U :coverage (know-domain :domain movies)))
(ii) QueU: Do you have the menus for restaurants?
(query U S :know-base (confirm S U :domain (describe-domain :domain $domain (has-feature :feature menu)))
(iii) S: There is one Persian restaurant in the Mid-Peninsula area.
QueU: Do you have the menu for it?
(query U S :database (confirm S U :member (has-attribute :member $member :feature menu)))
QueU: What are the opening hours for this restaurant?
(query U S :database (provide S U :member (describe-member :member $member :feature hours)))
(iv) QueS: Shall I remind you how we've narrowed down restaurants so far?
(query S U :task-info (suggest S U :space (describe-space :space $space-id)))
(v) QueU: What do you mean by restaurant type?
(query U S :know-base (provide S U :domain (describe-feature :domain $domain-id feature-list :feature rest-type :attribute range))

Table 3: Example Performatives for Queries

given constraints), only if the previous dialogue up to that point was not about searching the restaurant domain. The details of how natural language interpreter produces these messages are beyond the scope of this article. Suffice it to say that the abstract task model can play a large role in helping the natural language interpreter to achieve this, but this is not a focus of our concern in this paper.

We now have a representation and semantics for the abstract task model of successive refinement search and have covered all possible intentions (i.e., intentions in the sense of illocutions associated with speech acts) that agents can express when communicating about the search task. We cannot overemphasize the role of the abstract task model in specifying the ontology for these performatives. The successive-refinement task model defined a semantics for the general performatives in terms of what could be talked about. A more general theory of semantics for performatives in terms of what messages must or may follow each other derives from [15]. However, the content of the performatives, i.e., the objects of discourse and what can be said about them, can only follow from the joint commitment to a shared abstract task model. What we have developed here is an abstract task analysis for cooperative successive refinement search, and let that abstract task model define these objects, their relations, and the set of methods that allow progress to be made on the task.

5. ABSTRACT TASKS DEFINE INTENTIONS & GLOBAL COHERENCE

At this point, we have only covered utterances and their associated intentions in communication (i.e., the illocution of an utterance). Table 1's specifications about what performatives may or must follow each other ensure some local coherence at the level of 2-3 message sequences, but they do not structure how short message sequences can be patched together in a globally coherent way. We now consider how intentions are internalized for reasoning about the next course of action (progress on the task vs. progress in the discourse) and how to maintain a coherent dialogue between agents (i.e., coherent perlocutions). As for dialogues, we assume a turn-taking dialogue. The abstract task model we apply here allows (i) that both the agents can take the initiative in directing the task and (ii) the user's initiative always has priority over the system's initiative. While the user may change the direction or other aspects of the search at any time, the abstract task model defines that the system would do so only if the current direction could not be pursued further. In this case, the system takes initiative in suggesting a new course of action. The abstract task model, by specifying *strategies* for task progress (such as identifying the most discriminating feature to reduce a search set), also defines a larger set of intentions that the system can have and that can in turn be realized as suggestions to the user.

There must be (i) a means for deciding or recognizing that a particular message is relevant to advancing either the task, the exchange of information about the task, or switching to a completely new task, and (ii) a means for representing how the semantics of the performative pairs in Table 1 actually advance these goals. The first is accomplished by relating agent intentions about the task or the discourse to specific performatives. The second is accomplished by protocols, which explicitly map sequences of messages to the fulfillment of a particular intention, and indicate what must happen when an unexpected message is received. We now turn to a more detailed discussion of intentions and protocols, and how they are represented.

5.1. The Interplay of Task Space and Discourse Space

In previous sections, we have underscored that ATS must reason about both the task of successive refinement search and about the task of structuring the discourse with the user in an intelligent way. These two tasks are not only related, but at different times, one supports the other. Here, we introduce the conceptual distinction between "task space" and "discourse space", and indicate how intentions and protocols define elements of these two spaces.

Let a state in the discourse space correspond to shared information held by both ATS and the user. A transition between one state into another is made by making an utterance (e.g., "Where do you want to eat?"). When an utterance is correctly interpreted, its intention is understood and that intention becomes part of the shared information between the two agents. The coherence of the discourse is the coherence of an utterance sequence and this sequence will be coherent only if the utterance is interpreted correctly and responded to appropriately.

For the abstract task of successive-refinement search, the problem at the task level is to identify a set of entities in a database that meet certain constraints. A state in this task space is a set of database entities and the constraints that define them. There are two legal transitions or operators at this level: the contract operator, which means adding a new constraint, or the expand operator, which means relaxing an existing constraint. When either of these operators are applied within a particular state, a new state results, defined by a new set of database entities and the feature-value specifications that uniquely define them. The goal state in the task space is the set of database entities that the user deems to be sufficient for his or her purposes.

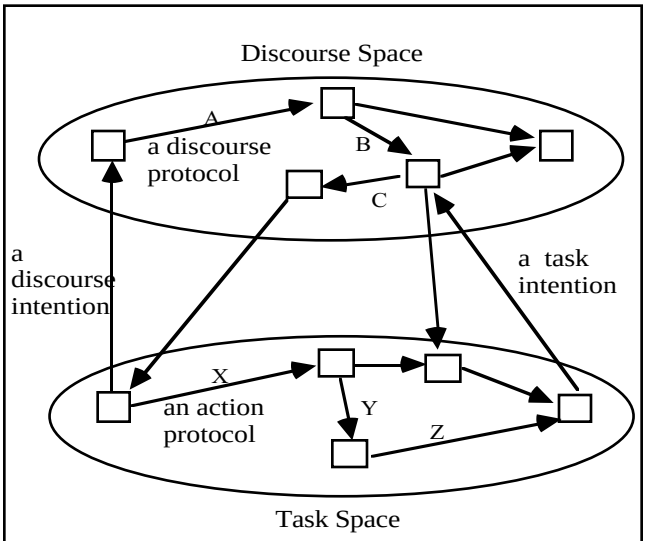

Figure 1: Relation between Discourse Space and Task Space

The interplay of task space and discourse space is summarized in Figure 1. The crucial feature is this: whenever an agent cannot proceed in one space, it must form an intention to make a transition in the other space. Thus, the failure to move ahead in the task space (e.g., contract the search space of entities) requires an intention to make a transition in the discourse space (e.g., a request for or suggestion about possible constraints). Conversely, a transition in the discourse space may require movement in the task space (e.g., a request to contract).

5.2 The Interplay of Intentions and Protocols

An intention to achieve a goal is represented as a general plan to advance either the task (in which case it is a *task intention*) or the exchange of information (in which case it is a *discourse intention*). Whenever the system cannot proceed in the task space, it adopts a discourse intention which causes the transition of control to discourse space until sufficient information has been gathered from the user to proceed in the task space (e.g., schematically illustrated as transitions ABC in Figure 1). Similarly, whenever the system cannot proceed further in the discourse space, a task intention is formed for a task computation that result in the information that can then be exchanged with the user by returning and proceeding in the discourse space (e.g., schematically illustrated as transitions XYZ in Figure 1).

An intention is represented as a plan consisting of (i) a *triggering condition*, which returns true if the intention is applicable to the current task context; (ii) a c*ompletion routine*, (iii) a *method-to-advance* routine; and (iv) a *resumption* routine. The completion routine checks if there is enough information to satisfy the intention. If there is, a transition is made in the current space by executing one of the protocols specified in the completion routine which is appropriate in the current task context. Otherwise, the method-to-advance routine determines a method for adopting an intention appropriate to the current task context in order to satisfy the completion condition. The adopted intention is generally an intention to move in the other space. Without entering into the architectural requirements for an ATS framework (see [9] for details), the current task context is determined by checking the value of relevant task context variables which hold sufficient information about the current state in the task space and in the dialogue space. Finally, the resumption routine indicates whether the intention, if it had been suspended, can be resumed. We will return to this later.

A *protocol* defines a structured exchange of information between two agents. Protocols are the methods by which an intention is either completed (its completion routine) or advanced (its method-to-advance routine). Just as there are two types of intentions, so too there are *discourse protocols* and *task protocols*. Task protocols govern the exchange of information between ATS's Dialogue Manager and what we term ATS's Task Manager that performs task-level computations such as retrieving information from databases, creating a search space and computing most discriminating features in the current space. For the remainder of this paper, however, we only focus on discourse protocols.

Discourse protocols are invoked by discourse intentions. The definition of protocols includes the specifications of the semantics for short sequences of performatives given in Table 1. They embody the rules that dictate what performatives must temporally follow others. Protocols can also be viewed as plans, whose representation is based on three critical elements: (i) *invocation condition*, which is the name of its invoking intention, (ii) *invocation routine*, which is executed upon invocation to create a message of the appropriate type and send it to the natural language generator, and (iii) *assimilation routine*, which awaits one or more performatives associated with the user's utterance.

It is in the definition of the assimilation routine that we allow for "violations" of the protocol, i.e., the arrival of a message that is unexpected. When a performative arrives that cannot be interpreted in the context of this protocol, the protocol and its invoking intention are suspended, handing over the user performative to ATS's higher level discourse management algorithm (see next section). There the user performative will trigger the recognition of a user intention. This is the manner in which ATS can switch to a state in the task space that is not a predefined transition from its current task state. ATS's modest amount of "reasoning" allows unexpected performatives (i) to be recognized as unexpected, (ii) to trigger the recognition of user intention and (iii) to pass control to responding to that intention. Exactly what new task intention is being signaled by the unexpected message requires a scheme for intention-matching on the part of ATS. But the crucial role of the abstract task model, in conjunction with the current task context, is to define a small set of possibilities.

A suspended intention may never be resumed, if the user intention signals a valid change of direction in the search task. Here is where the distinction we make between request and query performatives play an important role. Requests are directly associated with transitions in the task space and a commitment to making those transitions. Hence a user request causes adoption of a new task intention. If the completion condition of this task intention is true, the course of the current task context will change and this will be reflected in the task context variables. The resumption routine of a suspended intention will check these variables to determine if the intention may be resumed or abandoned. We require that a new direction in the search task results in abandoning a suspended task intention and its associated protocol. If the user performative, however, was a query, the suspended intention may be resumed. The resumption routine of that intention then checks the context variables to see what the next coherent course of action should be in the new situation.

5.4 Dialogue Management Algorithm

The interplay of task-level and discourse-level intentions, as produced and defined by the abstract task model, and the management of protocols that are called in service of these intentions is realized by a dialogue management algorithm. Since both intentions and protocols can be viewed as plans, the basic control algorithm can be viewed as passing control from plan to plan, until the task is completed. While local coherence is maintained within protocols, global coherence is maintained by the dialogue management control algorithm. On the abstract level, given the turn-taking assumption in dialogue, the algorithm follows the following priority scheme: (i) recognize the need to set or

respond to discourse intentions, (ii) react to any user intentions, and (iii) set and pursue ATS task intentions. ATS task intentions are set and pursued through the combination of an intention priority scheme (defined by the abstract task model) that indicates how features of the current task context trigger certain types on intentions. For example, if the search set size has become very small, an appropriate intention might be to suggest that one or more of its members be described to the user in more detail. To give a general flavor of the control scheme Figure 2 presents an abbreviated view of the dialogue management algorithm.

We note that this sort of dialogue (or conversation) management algorithm characterizes much of the work in the discourse processing area. What is new here is our reliance on formalizing an abstract task model to implement the crucial aspects of run-time intention recognition in a practical and domain independent way.

Until dialogue is over	
If	there are user performatives posted in the discourse context
Then	map these to either a user task-intentions or a user discourse-intentions
elseif	there is a current intention that can be completed
then	complete it & remove it as current & pop the intention stack
elseif	there is a current intention that had been suspended
then	check its resumption conditions & either resume it and its associated protocol, or abandon it and its associated protocols and any intentions for which it was a necessary subgoal
elseif	there is a current intention that can not be satisfied
then	transfer control to its "method to advance", triggering a discourse protocol with the other agent
elseif	there is no current intention
then	set a task intention by examining the task context

Figure 2: Overview of Discourse Management Algorithm

7. DISCUSSION

The framework we have described could be viewed as a two-level state-transition model: the abstract task model defines transitions at the level of task intentions and the discourse protocols define transitions at the level of shorter sub-conversation messages. Task intentions essentially structure the problem solving process and in turn, the dialogue that can ensue about the problem solving process. However, we allow that certain task intentions may be forever abandoned, as the initiative between agents shifts during the problem solving process. Hence, we allow for transitions as well as "jumps" between intentional states. It is through this sort of approach that we aim to realize a pragmatic implementation of notions such as beliefs and intentions. While mixed initiative dialogues are currently mostly studied in relation to interacting with users, we envision mixed initiative dialogues between software agents to be an important requirement for many applications systems of the future. In particular, in complex settings such as agent to agent negotiation, where protocols as they are typically employed simply cannot provide sufficient flexibility needed by agents for exploring the negotiations space.

One matter for further consideration is how the idea of an abstract task model fits with the decomposition of a complex task into a set of subtasks.[3] One issue is whether the notion of a protocol "violation" is too restrictive a mechanism for signaling intention changes. When a task consists of a series of subtasks, any message that signals a move to the next subtask does not seem like a "violation" but rather, a natural and well-defined transition. We do not intend that protocol violations are the only means of shifting attention to a new intention. In our brief discussion of the dialogue management algorithm, we noted that the algorithm's last step is "set and pursue intentions" that are appropriate to the current problem status. Those candidate intentions in turn are specified within the abstract task model and they can be viewed as a set of subtasks (or more precisely, they are a set of problem situations about which the agent can have intentions). In our case, these task situations and associated intentions are quite simple: (1) as long as the search space is very large, adopt intentions to make it more constrained; (2) if the search space is empty, adopt intentions to relax constraints; (3) if the search space is smallish (however defined), identify one or more members to describe in detail. So these sorts of subtasks, and their associated intentions, are determined by the abstract model and constitute something like a plan of attack. A hierarchical task network planning approach is likely to provide further structure for very complex tasks; at this point, we do not see any incompatibility between a hierarchical specification of tasks within an abstract task model, and the general benefit we see for taking an abstract task model approach. We use protocol violations merely as a way to signal whether another intention defined within the task model might be in play. If the agents are "between" protocols (and possibly between subtasks), then there is no sense in which any incoming message is a violation.

A second issue concerns the dominance of the user in the approach we have advocated here. On the one hand, we are calling this a 'mixed initiative approach', which makes it seem as if both agents have equal status. On the other hand, the ATS's intentions take back seat to any user intention. Note that this is defined strictly within the discourse management algorithm and not within the abstract task model. If both agents are software agents, then the question naturally arises about which agent steers the conversation, or whether they both would be searching inefficiently for sub-conversations. This is a very important concern to

[3] We acknowledge one of the reviewers for raising the issues about subtasks, hierarchical task network planning, and user dominance that we consider in these paragraphs.

raise, for it spotlights an important matter to revisit, if not solve. Two agents have to cooperate because they each have access to different sorts of knowledge and different capabilities. Even if the two agents shared the same abstract task model (and by definition, this would include the same strategy or plan of attack outlined in the previous paragraph), one of them must know something or be able to do something the other cannot. (In our case, the human agent has the constraint information and the goal information, and the ATS agent has the database access and system functionality to compute the results). If these were two software agents, there would have to be some exchange concerning "what do I know" vs. "what do you know" to define the nature of the cooperation. Who-knows-what might determine which agent's intentions have priority during the conversation. For example, the agent who possesses the definition of the goal state might have some kind of intention advantage. This is not a function of the abstract task model, but of the discourse algorithm that uses the model. We note that our remarks here are just speculations, for this matter involves knowledge brokering among agents, which this work does not address. We presume that one agent will have the role of declaring a goal state has been reached; if the agents themselves must first agree on what would constitute a goal state, that is yet another matter outside the scope of this work.

We are presently implementing the ATS design presented here in a PRS framework, which fits well with our view of intentions and protocols as plans. The critical, more general issue concerns how to assess the pragmatic advantage that may be gained by specifying the semantics for abstract tasks as the semantics that are required to support flexible multiagent communication. Within the knowledge-based system arena, Chandrasekaren and his colleagues advanced the notion that complex tasks could be decomposed into sub-tasks that were, in some sense, generic, such as classification, data retrieval, plan selection and refinement, and so forth. If there are generic ontologies associated with generic tasks, then those ontologies can be the foundation for intentions that, in turn, can structure the conversation and the content of the individual performatives. Thus, a crucial area for further work is assessing whether and how this abstract (or generic) task approach can be leveraged to define semantics for a core set of objects-of-discourse to be used in the evolution of some standard task languages for agents.

8. ACKNOWLEDGMENTS

Aspects of this work were developed while A. Haddadi was seconded to the Adaptive Systems Laboratory, Daimler-Benz Research and Technology Center, Palo Alto, California, where R. Elio spent part of a sabbatical during 1997-1998. The support of that group, the University of Alberta, and NSERC Grant A0089 to R. Elio is gratefully acknowledged. We also thank two anonymous reviewers for their comments and suggestions, which helped to improve the presentation of this work.

9. REFERENCES

[1] Allen, J. F., Schubert, L.K., Ferguson, G. Heeman, P., Hwang, C.H., Kato, T., Light, M., Martin, N. G., Miller, B. W., Poesio, M., Traum, D. (1995). The TRAINS Project: A case study in building a conversational planning agent. *Journal of Experimental and Theoretical AI* 7: 7-48.

[2] Austin, J. (1956). *How to do things with words*. Oxford: Oxford University Press.

[3] Bradshaw, J. M., Dutfield, S., Benoit, P., & Woolley, J. D. (1997). KAoS: Toward an industrial-strength open agent architecture. In J. M. Bradshaw (Ed.), *Software Agents*, 375-418. Menlo Park, CA: AAAI Press.

[4] Bylander, T. & Chandrasekaren, B. (1987). Generic tasks for knowledge-based reasoning: The "right" level of abstraction for knowledge engineering. *International Journal of Man-Machine Studies* 26: 231-243.

[5] Chandrasekaren, B. & Johnson, T. R. (1993). Generic tasks and task structures: History, critique, and new directions. In D. Krivine & R. Simmons (Eds.), *Second Generation Expert Systems*, 232-272. Berlin: Springer-Verlag.

[6] Cohen, P. R. & Levesque, H. J. (1997). Communicative actions for artificial agents. In J. M. Bradshaw (Ed.), *Software Agents*, 419-436. Menlo Park, CA: AAAI Press.

[7] Cohen, P. R. (1994). Models of dialogue. In T. Ishiguro (Ed.) Cognitive Processing for Vision and Voice: *Proceedings of the Fourth NEC Research Symposium*. 181-203. Philadelphia: Society for Industrial and Applied Mathematics.

[8] Cohen, P.R. & Levesque, H. J. (1990). Intention is choice with commitment. *Artificial Intelligence* 42: 213-262.

[9] Elio, R. & Haddadi, R. (1998). Dialogue management for an adaptive database assistant. DaimlerBenz North America Research and Technology Center Report 3-98, Palo Alto, CA.

[10] Ferguson. G., Allen, J.F., & Miller, B. (1996). TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third Conference on Artificial Intelligence Systems (AIPS-96)*, 70-77. Menlo Park, CA: AAAI Press.

[11] Genesereth, M.R. (1997). An agent-based framework for interoperability. In J. M. Bradshaw (Ed.) *Software Agents*. Menlo Park, CA: AAAI Press.

[12] Grice, H.P. (1975). Logic and conversation. In P. Cole & J.L. Morgan (Eds.) *Syntax and Semantics*, Vol 3: Speech Acts, 225-242. New York: Academic Press.

[13] Grosz, B.J. & Sidner, C.L. (1986). Attention, intention, and the structure of discourse. *Computational Linguistics* 12:175-204.

[14] Haddadi, A. (1998) Towards a pragmatic theory of interactions. In M. N. Hinghs & M. P. Singh (Eds.) *Readings in Agents*, 443-449. San Francisco: Morgan Kaufmann.

[15] Haddadi, A. (1996). *Communication and cooperation in agent systems: A pragmatic theory*. Lecture Notes in Computer Science, #1056. Berlin: Springer Verlag.

[16] Labrou, Y. & and Finin, T. (1994). A semantics approach for KQML—A general purpose communication language for software agents. In *Third International Conference on Information and Knowledge Management*, 47-455. New York: ACM Press.

[17] Labrou, Y. & Finin, T. (1998). Semantics and conversations for an agent communication language. In M. N. Hinghs & M. P. Singh (Eds.) *Readings in Agents*, 235-242. San Francisco: Morgan Kaufmann.

[18] Mayfield, J., Labrou, Y., & Finin, T. (1996). Evaluation of KQML as an agent communication language. In M. Woodbridge, J. P. Muller, & M. Tambe (Eds.) *Intelligent Agents II: Agent Theories, Architectures, and Languages*: Lecture Notes in Artificial Intelligence, # 1037.Berlin: Springer-Verlag.

[19] Reinhart, T. (1981). Pragmatics and linguistics: An analysis of sentence topics. *Philosophica* 27: 53-94.

[20] Rich, C. & Sidner, C. L. (1998). COLLAGEN: When agents collaborate with people, In M. N. Hinghs & M. P. Singh (Eds.) *Readings in Agents*, 117-124, San Francisco: Morgan Kaufmann.

[21] Searle, J.R. (1969). *Speech Acts*. New York: Cambridge University Press.

[22] Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence* 60: 51-92.

[23] Singh, M. P. (1998). Agent communication languages: Rethinking the Principles. *IEEE Computer* 31: 40-49.

[24] Winograd, T. & Flores, F. (1987). *Understanding computers and cognition*. New Jersey: Ablex.