A Problem Solving Model for Collaborative Agents

James Allen
Dept. of Computer Science
University of Rochester
Rochester, New York, USA
james@cs.rochester.edu

Nate Blaylock
Dept. of Computer Science
University of Rochester
Rochester, New York, USA
blaylock@cs.rochester.edu

George Ferguson
Dept. of Computer Science
University of Rochester
Rochester, New York, USA
ferguson@cs.rochester.edu

ABSTRACT

This paper describes a model of problem solving for use in collaborative agents. It is intended as a practical model for use in implemented systems, rather than a study of the theoretical underpinnings of collaborative action. The model is based on our experience in building a series of interactive systems in different domains, including route planning, emergency management, and medical advising. It is currently being used in an implemented, end-to- end spoken dialogue system in which the system assists a person in managing their medications. While we are primarily focussed on human-machine collaboration, we believe that the model will equally well apply to interactions between sophisticated software agents that need to coordinate their activities.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Natural Language*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Theory and Methods*

Keywords

Conversational agents, interface agents, coordinating multiple agents and activities, intention recognition

1. INTRODUCTION

One of the most general models for interaction between humans and autonomous agents is based on natural human-human dialogue. For humans, this is an interface that requires no learning, and provides maximum flexibility and generality. To build such an interface on the autonomous agent side, however, is a formidable undertaking. We have been building prototypes of such systems for many years, focusing on limited problem solving tasks. Our approach involves constructing a dialogue system that serves as the interface between the human and the back-end agents. The goal is to insulate the human from the complexities of managing and understanding agent-based systems, while insulating the back-end agents from having to understanding natural language dialogue. To be effective in a range of situations, the dialogue agent must support contextually-dependent interpretation of language and be able to map linguistically specified goals into concrete tasking of back-end agents.

We believe that a key for enabling such interaction models is the development of a rich model of collaborative problem solving. This model is needed for two distinct purposes: (1) to enable contextual interpretation of language (*i.e.*, intention recognition); and (2) to provide a rich protocol for communication between the autonomous agents that comprise the dialogue system. Thus the dialogue system appears to the human as an intelligent collaborative assistant agent, and is itself comprised of autonomous agents.

While work has been done on general theoretical frameworks for collaborative interaction [8, 3, 11], these proposals have generally not specified the details of what such models would look like. We believe that our model is compatible with the SharedPlans formalism [8, 9, 11]. In fact, one way of looking at our model is as an elaboration of some of the key operators (such as Elaborate_Group, or Lochbaum's *communicate* recipe) in the SharedPlans framework. In our own previous work [6, 1], we have described the beginnings of practical models but these have not been very precisely specified or complete. In this paper, we sketch a comprehensive model that provides a detailed analysis of a wide range of collaborative problem solving situations that can arise. This model is based on our experience in building collaborative problem solving agents in a range of different domains. In particular, collaborative agents (both human and autonomous) need to have the capability to:

1. Discuss and negotiate goals;
2. Discuss options and decide on courses of action, including assigning different parts of a task to different agents;
3. Discuss limitations and problems with the current course of action, and negotiate modifications;
4. Assess the current situation and explore possible future eventualities;
5. Discuss and determine resource allocation;
6. Discuss and negotiate initiative in the interactions;

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

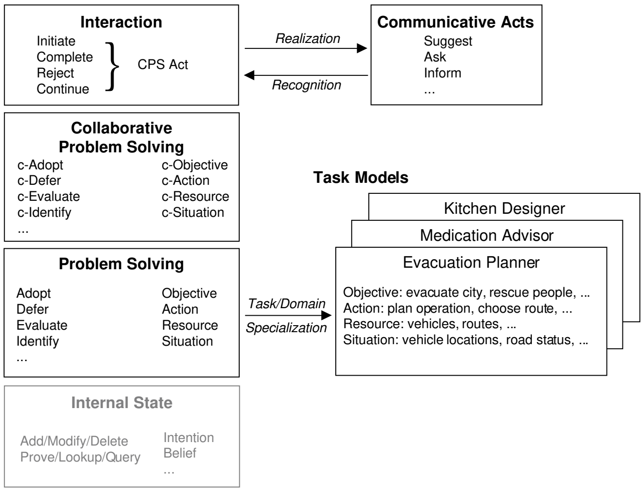

Figure 1: Collaborative problem solving model

7. Perform parts of the task, and report to others to update shared knowledge of the situation.

Although our focus is on language-based interaction, it is our belief that these capabilities are required in any sufficiently complex (realistic, flexible) agent-based system.

2. OVERVIEW OF THE MODEL

Our model of collaborative problem solving is shown in Figure 1. At the heart of the model is the **problem solving level**, which describes how a single agent solves problems. For example, an agent might adopt an obligation, or might evaluate the likelihood that a certain action will achieve that objective. This level is based on a fairly standard model of agent behavior, that we will describe in more detail shortly.[1]

The problem solving level is specialized to a particular task and domain by a **task model**. The types of domains we have explored include designing a kitchen, providing medical advice, assessing damage from a natural disaster, planning emergency services, and so on. The task model describes how to perform these tasks, such as what possible objectives are, how objectives are (or might be) related, what resources are available, and how to perform specific problem solving actions such as evaluating a course of action.

For an isolated autonomous agent, these two levels suffice to describe its behavior, including the planning and execution of task-level actions. For collaborative activity, however, we need more.

The **collaborative problem solving level** builds on the single-agent problem solving level. The collaborative problem solving actions parallel the single-agent ones, except that they are joint actions involving jointly understood objects. For example, the agents can jointly adopt an intention (making it a joint intention), or they can jointly identify a relevant resource, and so on.

[1]Underlying the problem solving level is the representation of the agent's internal state, for example its current beliefs and intentions. The details of how these are represented are not important for understanding the collaborative problem solving model, however.

Finally, an agent cannot simply perform a collaborative action by itself. The **interaction level** consists of actions performed by individuals in order to perform *their* part of collaborative problem solving acts. Thus, for example, one agent may initiate a collaborative act to adopt a joint intention, and another may complete the collaborative act by agreeing to adopt the intention.

This paper proceeds as follows. First, we describe the collaborative problem solving model in more detail, starting with a review of some underlying concepts, moving on to the single-agent problem solving level, and finally describing the collaborative problem solving and interaction levels. The emphasis is on the information maintained at each level and its use during collaborative problem solving. We then present a detailed example of the model in action, drawn from a medical advisor domain that we are using for our prototype implementation. We conclude with a few comments about open issues and future work.

3. BASIC CONCEPTS

All of the levels in our model involve a core set of concepts related to planning and acting. Many of these concepts have been used in the planning literature for years, and we only informally describe them in this section. The application of the concepts to modeling collaborative interaction is what is important for present purposes.

3.1 Situations

We start with a fairly standard notion of **situation** as in the situation calculus [12]—a situation is a snapshot of the world at a particular point in time (or hypothetical point in time when planning into the future). While situations are a complete state of the world at a certain time, our knowledge of a situation is necessarily incomplete except in the most simple cases (like traditional blocks world planning systems). Also note that a situation might include an agent's beliefs about the past and the future, and so might entail knowledge about the world far beyond what is immediately true.

3.2 Atomic Actions

Also as in the situation calculus, **actions** are formalized as functions from one situation to another. Thus, performing an action in one situation produces a new situation. Of course, generally we do not know the actual situation we are in, so typically knowledge about actions is characterized by statements that if some precondition of an action is true in some situation, then some effect of it will be true in the situation resulting from the action. Note that unlike the standard situation calculus, however, we take actions to be extended in time and allow complex simultaneous and overlapping actions.

3.3 Recipes

A specification of system behavior is often called a plan, or **recipe** [13]. We will use the term "recipe" here as the notion of plan has been overused and so is ambiguous. A very simple form of recipe is a fixed sequence of actions to perform, much like those built by traditional planning systems. The recipes found in cookbooks often aspire to this level of simplicity but typically are not as straightforward. More generally, recipes capture complex learned behavior and guide an agent towards a goal through a wide range of possible conditions and ranges of possible results from previous action.

For our work, we do not care about the specific form of what a recipe is, or insist that different agents have the same recipes. Rather, a recipe is a way of deciding what to do next. More formally, a recipe is a function from situations to actions, where the action is the next thing to do according to the recipe.

Note that we need some special "actions" to make this work. First, we must allow the action of doing nothing or waiting for some period of time, as this might be the best thing to do for some recipes. We also need to allow the possibility that a recipe may not specify what to do next in certain situations. To formalize this, we need to make the recipe function a partial function, or introduce a special "failure" value. Finally, we need to allow actions to be planning actions—*i.e.*, it may be that the best thing to do is to set a subgoal and do some more planning before any further physical action.

3.4 Objectives

Our notion of **objective** is similar to some uses of the term "goal." But the term goal is used is different ways in the literature: goals are sometimes the intentions driving an agent's behavior, at other times they are the input to a planning process, and sometimes they are simply the main effects of a recipe. Goals are sometimes considered to be states of the world to attain (*e.g.*, the goal is a situation where block A is on block B), or sometimes an action that must be performed (*e.g.*, the goal is to open the door).

We will try to avoid all this ambiguity by not using the word goal any further. An **objective** is an intention that is driving our current behavior. Objectives are expressed in the form of abstract actions, such as winning the lottery, or getting block A onto block B. Objectives are not just any actions. They are actions that are defined in terms of their effects, and cannot be executed directly. To accomplish objectives, we need to choose or build a recipe that, if followed, leads to a state in which effects of the objective hold.

3.5 Resources

The final key concept in the abstract model is that of a **resource**. A resource is a object that is used during the execution of a recipe. Resources might be consumable (*i.e.*, cease to exist in their prior form) as a result of the recipe (*e.g.*, as ingredients are consumed when making a cake), or might be reusable (*e.g.*, as a hammer is used to drive in a nail). In a traditional planning model, resources are the objects that are used to bind the variables in the plan and, in fact, many applications of planning are essentially resource allocation problems.

4. PROBLEM SOLVING

Once we have the concepts defined in the last section, we can now give an quick overview of our model of a single agent's problem solving behavior. Just as task-level actions affect the state of the world, problem-solving actions affect the cognitive state of the agent, which we represent (for purposes of this paper) as the **problem solving (PS) state**. The problem solving state consists of the agents commitments towards objectives, the recipes for achieving those objectives, the resources used in those recipes, and so on.

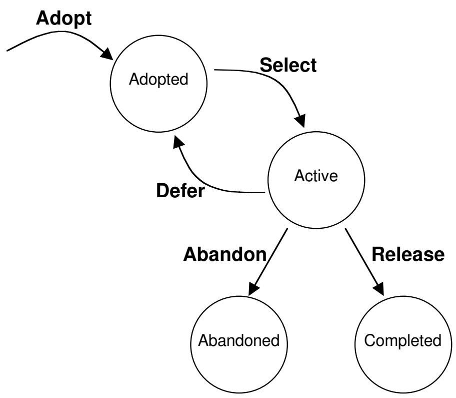

Figure 2: Life cycle of an intention

The PS state must contain at least the following information:

1. The current situation: what the agent believes (or assumes) to be true as a basis for acting, including what resources are available;

2. Intended objectives: a forest of objectives that the agent has adopted, although not all are necessarily motivating its current action. Each tree in the forest captures a subobjective hierarchy for a particular root objective;

3. Active objective(s): the intended objective(s) that is (are) currently motivating the agent's action; An objective tree that includes an active objective is an active objective tree;

4. Intended recipes: the recipes and resources that the agent has chosen for each of the intended objectives;

5. Recipe library: a set of recipes indexed by objective and situation types. The library need not be static and may be expanded by planning, learning, adaptation, *etc.*

An agent's problem solving activity involves exploring the current situation, adopting objectives, deciding on courses of action and resources to use, performing actions, and using the results of actions to further modify its objectives and future actions. The problem solving actions (see Figure 1) are divided into two classes, one concerned with intention and commitment and one concerned with knowledge and reasoning.

4.1 PS Acts Relating to Commitment

In our model of agent behavior (similar to [7, 15]), an agent is driven by its intentions, in the form of objectives, recipes, and resource uses to which it is committed. Intentions move through a life cycle shown in Figure 2.

In order to act, an agent must form intentions by means of a commitment act that we call **Adopt**. For example, an agent might adopt a particular recipe for a certain objective, say to plan a trip by using a recipe to call a travel agent. If they change their mind, they may drop the commitment using a act we call **Abandon**. For instance, the agent may change their mind, abandon the recipe to call the travel agent and adopt a recipe to book a ticket on the web. Similarly, an agent may adopt or abandon objectives, and adopt and abandon commitments to use certain resources.

An agent may have several different objectives that it is committed to, and even with respect to one objective, there may be several sub-objectives that could be chosen to drive the agent's action. The action of choosing the objective(s) to motivate the next behavior is called **Select**. Once an objective is selected, the agent may perform reasoning to elaborate on its associated recipe, or to evaluate an action that that recipe suggests, and may eventually select an action to perform. If an agent's priorities change, it may **Defer** the objective or action, leaving it as an intention to be addressed later. Finally, when an agent believes that an objective has been achieved, it may **Release** the objective, thereby removing it from its set of intentions.

4.2 PS Acts Relating to Reasoning

Before committing, an agent will typically perform some reasoning. One of the key operations is to determine what options are available, which we call **Identify**. For example, an agent may identify a possible recipe for achieving some objective, or identify certain resources that are available. They may even identify possible goals to pursue and consider them before making any commitment. Once an option is identified, the agent may **Evaluate** it relative to its purpose. For instance, it might evaluate a recipe to see how well it might accomplish its associated objective, or evaluate an objective to see if it is worthwhile, or evaluate a resource or action to see how well it serves a particular recipe. In addition, an agent may choose to **Modify** a certain objective, recipe or resource to produce a another that then could be evaluated.

In addition to reasoning about possible goals and actions, an agent may also reason about its current situation. Situations may be identified by exploring them further, and may be evaluated to see how desirable the current (or expected) situation is and whether it should plan to change it. Agents that act and do little planning would only care about the current situation they are in, and all activity would be tied to that situation. More complex agents, however, could do planning in hypothetical situations, or want to act based on certain assumptions.

4.3 Problem Solving Behavior

With these elements of the problem solving model in place, we can describe how an agent solves problems. It is convenient to present this activity as occurring in a series of phases. In practice, an agent may short circuit phases, or return to prior phases to reconsider their commitments at any time.

1. *Determining the Objective:* An agent may at any time reconsider the objectives it has, adopt new ones, abandon old ones, and otherwise modify and adjust them. Of course effective agents will not spend too much time reconsidering and evaluating their objectives, but will spend their effort in pursuing an objective. To do this, they must first select one or more objectives to pursue. These are the active objectives.

2. *Determining the Recipe:* Given an active objective, an agent must then determine a recipe to follow that may achieve the objective. It may be that a recipe has already been used for some time to determine the agent's actions in pursuing the objective, and the agent may simply invoke the recipe once again in the current situation to determine what to do next. But the agent might also consider switching to another recipe, refining an existing recipe, or actually building a new recipe for this objective. In these latter cases, the next action the agent does is a planning action that results in a modified (or new) recipe for the objective.

3. *Using the Selected Recipe:* Given a selected recipe, the agent can then identify the next action to perform. If the recipe returns a sub-objective, then the agent needs to restart the process of evaluating objectives and choosing or constructing recipes. If the recipe indicates an atomic action, the agent can evaluate the desirability of the proposed action, and if it seems reasonable, perform the action. At that point, the situation has changed and the process starts again.

To implement such a problem solving agent, we would need to specify strategies for when objectives, recipes and proposed actions are evaluated and reconsidered, versus how often the current objective, recipe or proposed action is just taken without consideration. Agents that performed more evaluation and deliberation would be more careful and might be able to react better to changing situations, whereas agents that do less evaluation would probably be more responsive but also more brittle. The specifics of these strategies are not the focus of this paper.

5. COLLABORATIVE PROBLEM SOLVING

We now turn to the central issue of collaborative problem solving. When two agents collaborate to achieve goals, they must coordinate their individual actions. To mirror the development at the problem solving level, the collaborative problem solving level (see Figure 1) operates on the collaborative problem solving (CPS) state, which captures the joint objectives, the recipes jointly chosen to achieve those objectives, the resources jointly chosen for the recipes, and so on.

The collaborative problem solving model must serve two critical purposes. First it must provide the structure that enables and drives the interactions between the agents as they decide on joint objectives, actions and behavior. In so doing, it provides the framework for intention recognition, and it provides the constraints that force agents to interact in ways that maintain the collaborative problem solving state. Second, it must provide the connection between the joint intentions and the individual actions that an agent performs as part of the joint plan, while still allowing an agent to have other individual objectives of its own.

While we talk of shared objectives, intended actions and resources, we do not want to require that agents have the same library of recipes to choose from. This seems too strong a constraint to place on autonomous agents. We assume only that the agents mutually agree on the meaning of expressions that describe goals and actions. For example, they might both understand what the action of taking a trip entails. The specific recipes each has to accomplish this action, however, may be quite different. Their recipes may accomplish subgoals in different orders for instance (one may book a hotel first, then get a air ticket, where the other might reverse the order). They might break the task down into different subgoals (*e.g.*, one may call a travel agent and book flight and hotel simultaneously, while the other might book flights with an agent and find hotels on the web). And for any subgoal, they might pick different actions (*e.g.*, one might choose a flight that minimizes cost, whereas the other might minimize travel time). To collaborate, the agents must agree to some level of detail on a new abstract joint recipe that both can live with. The joint recipe needs be refined no further in places where the two agents agree that one agent is responsible for achieving a sub-objective.

Establishing part of the collaborative problem solving state requires an agreement between the agents. One agent will propose an objective, recipe, or resource, and the other can accept , reject or produce a counterproposal or request further information. This is the level that captures the agent interactions. To communicate, the agent receiving a message must be able to identify what CPS act was intended, and then generates responses that are appropriate to that intention. In agent-communication languages between programs, the collaborative act would be explicit. In human-agent communication based on natural language, a complex intention recognition process may be required to map the interaction to the intended CPS act. This will be described in further detail in the Interaction section below, after the abstract collaborative model is described.

5.1 Collaborative Problem Solving Acts

As a first cut, the collaborative problem solving level looks just like the PS level, except that all acts are joint between the collaborating agents. We will name these CPS acts using a convention that just applies a prefix of "c-". Thus the *c-adopt-objective* act is the action of the agents jointly adopting a joint objective.

While we can model an individual agent adopting an individual objective as a primitive act in our model at the PS level, there is no corresponding primitive act for two agents jointly adopting a goal. This would require some sort of mind synchronization that it not possible. We agree with researchers such as Grosz and Sidner [8] and Cohen and Levesque [3] in that joint actions must be composed out of individual actions. There remains a meaningful level of analysis that corresponds to the PS level model if we view the CPS acts as complex acts, *i.e.*, objectives, that the agents recognize and use to coordinate their individual actions. The constraints on rational behavior that an agent uses at the PS level have their correlates at the collaborative PS level, and these inform the intention recognition and planning behavior of the agents as they coordinate their activities. For instance, a rational individual agent would not form an objective to accomplish some state if it believed that the state currently holds (or will hold in the future at the desired time). Likewise, collaborating individual agents would not form a collaborative objective to achieve a state that they jointly believe will hold at the (jointly) desired time. The analysis of the behavior at this abstract level provides a simple and intuitive set of constraints on behavior that would be hard to express at the interaction action level.

5.2 The Interaction Level

The interaction level provides the connection between the communicative acts (*i.e.*, speech acts) that the agents perform, such as requesting, informing, warning, and promising, and the collaborative problem solving acts they jointly perform. In other words, it deals with the individual actions that agents perform in order to engage in collaborative problem solving. All the acts at this level take the form of some operator applying to some CPS act. For instance, an agent can **Initiate** a collaborative act by making a proposal and the other agent can **Complete** the act (by accepting it) or **Reject** it (in which case the CPS act fails because of lack of "buy in" by the other agent). In more complex interactions, an agent may **Continue** a CPS act by performing clarification requests, elaborations, modifications or counter-proposals. The interaction-level acts we propose here are similar to Traum's [17] grounding act model, which is not surprising as grounding is also a form of collaborative action.

From a single agent's perspective, when it is performing an interaction act (say, initiating adoption of an joint objective), it must plan some communicative act (say, suggesting to the other agent that it be done) and then perform (or *realize*) it. On the other side of the coin, when one agent performs a communicative act, the other agent must *recognize* what interaction act was intended by the performer. Identifying the intended interaction acts is a critical part of the intention recognition process, and is essential if the agents are to maintain the collaborative problem-solving state. For instance, consider a kitchen design domain in which two agents collaborative to design and build a kitchen. The utterance "Can we put a gas stove beside the refrigerator" could be said in order to (1) ask a general question about acceptable practice in kitchen design; (2) propose adding a stove to the current design; or (3) propose modifying the current design (say by using a gas stove rather than an electric one). Each of these interpretations requires a very different response from the hearer and, more importantly, results in a different situation for interpreting all subsequent utterances. Each one of these interpretations corresponds to a different collaborative problem solving act. If we can identify the correct act, we then have a chance of responding appropriately and maintaining the correct context for subsequent utterances.

We should note that the interaction level is not just required for natural language interaction. In other modalities, the same processes must occur (for example, the user initiates a joint action by clicking a button, and the system completes it by computing and displaying a value). In standard agent communication languages, these interaction level actions are generally explicit in the messages exchanged between agents, thereby eliminating the need to recognize them (although not to the need to understand and perform them oneself).

5.3 Examples

To put this all together, consider some typical but constructed examples of interactions. These examples are motivated by interactions we have observed in a medical advisor domain in which the system acts to help a person manage their medications. These examples are meant to fit together to form a constructed dialogue that illustrates a number of points about the CPS level analysis.

The simplest collaborative acts consist of an initiate-complete pair. For example, here is a simple *c-identify* of a *situation*:

U: Where are my pills? (1)
S: In the kitchen (2)

Utterance (1) is a Wh-question that initiates the *c-identify-situation* act, and utterance (2) answers the question and completes the CPS act.[2] When utterance (2) is done, the two agents will have jointly performed the *c-identify-situation* action.

Utterances may introduce multiple collaborative acts at one time, and these may be completed by different acts. For instance:

S: It's time to take an aspirin (3)
U: Okay (4)
U: [Takes the aspirin] (5)

Utterance (3) is a suggestion that U take an aspirin, which initiates both a *c-adopt-objective* (to intend to take medication currently due) and a *c-select-action* (to take an aspirin). Utterance (4) completes the *c-adopt* action and establishes the joint objective. Action (5) completes the *c-select* action by means of U performing the PS-level act *select* on the action, resulting in the action being performed.

Many more complex interactions are possible as well. For instance:

U: What should we do now? (6)
S: Let's plan your medication for the day (7)
U: Okay (8)

Utterance (6) is a question that initiates a *c-adopt-objective*, utterance (7) continues this act by answering the question with a suggestion, and utterance (8) completes the act (thus establishing the joint objective). Note that the objective agreed upon is itself a collaborative problem solving act—they have established a joint objective to perform a *c-adopt* action for some as yet unspecified *recipe*. This could then lead to pursuing a sub-objective such as creating a recipe as in the following interaction:

S: You could take your celebrex at noon. (9)
U: Will that interfere with my lunch date (10)
S: No. (11)
U: OK. I'll do that (12)

Utterance (9) is a suggestion that initiates a *c-identify-recipe* and continues the previously established *c-adopt-objective* action. Utterance (10) completes the *c-identify* of the recipe (by grounding the suggestion), continues the *c-adopt* action, and initiates a *c-evaluate* of the recipe by exploring a possible problem with the suggested action. Utterance (11) completes the *c-evaluate* act by answering the question, and utterance (13) then completes the *c-adopt* act by agreeing to the recipe initially suggested in (9).

6. EXTENDED EXAMPLE

To better illustrate the complexity of even fairly simple collaborative problem solving, the following is an example of a session with a prototype Medication Advisor system under development at Rochester [5]. The Medication Advisor is designed to help people manage their prescription medication regimes—a serious real-world problem that has a significant impact on people's health.

[2]Note that we are ignoring grounding issues in this paper. In a dialogue system, the CPS act is not actually completed until the answer to the question is grounded by U, say by the utterance such as "OK" or "thanks".

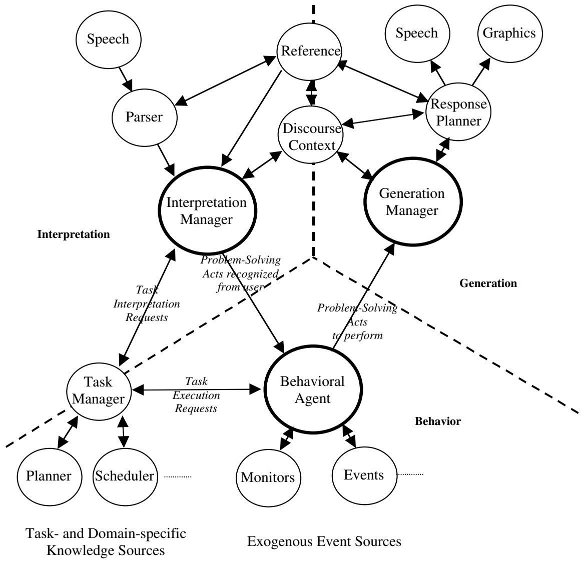

Figure 3: TRIPS collaborative system architecture (from [1])

To follow the problem solving, we need to understand something of the architecture of the system. The Medication Advisor is an application of the TRIPS spoken dialogue system [6], whose architecture is shown in Figure 3. As described in [1], the main components of the system as regards problem solving are as follows:

- The Interpretation Manager (IM), which maintains the discourse context and recognizes user intention from their utterances;
- The Behavioral Agent (BA), which manages system problem solving obligations and drives system behavior;
- The Task Manager (TM), which maintains the collaborative problem solving state and supports both interpretation and execution with task- and domain-specific knowledge; and
- The Generation Manager (GM), which coordinates the generation of spoken speech and multimodal output, among other functions.

This system is a fully implemented, end-to-end speech-driven dialogue system, although for this session some aspects of the problem-solving model were not fully implemented. For brevity, we have suppressed most of the detail of the actual message traffic between components of the system, focusing solely on the interactions within the collaborative problem solving model. Utterances are labeled by the agent and a number indicating the sequence of speaker turns.

The user starts by asking a question:

U: Can I take an aspirin? (1)

The Interpretation Manager asks the Task Manager to interpret the utterance in the current problem solving context. The Task Manager replies that the utterance looks like an *initiate* of a *c-evaluate-action*. That is, the user seems to be initiating a collaborative evaluation of the action of taking aspirin. The Interpretation Manager uses this result, together with other interpretation steps, and broadcasts that as the system's understanding of what the user has just done.

When the Task Manager receives this message, it can update its state to include the fact that, yes, the user has initiated a new collaborative act. Meanwhile, the Behavioral Agent also receives the notification. It decides, based on its current obligations, that the system should attempt to complete the action. To do this, it asks the Task Manager to perform the (non-collaborative) act of evaluating an action. The Task Manager does this by following a recipe for evaluating that type of action, which involves querying databases for what medications the user is taking and whether any of those interact with aspirin. It turns out that the there is an interaction, so the evaluation is negative. The Behavioral Agent, upon receipt of the result of the evaluation, requests that the Generation Manager perform the communicative act *inform* (with the evaluation results) in order to realize the interaction act of completing the user's *c-evaluate-action* CPS act.

The Generation Manager determines what to say:

S: No, you are taking celebrex and celebrex interacts with aspirin. (2)

Once the speech has been generated, the Generation Manager can broadcast to the rest of the system that the *inform* act is done. Note that the interaction act (the *complete*) is not yet done, since this depends on the user having understood the system's utterance (*i.e.*, on its being grounded).

The user responds with:

U: Oh. (3)

The Interpretation Manager takes this as grounding its utterance (2), since it does not indicate any lack of understanding. This fact is broadcast to the rest of the system. Once the system's utterance is grounded, the Generation Manager can announce that the interaction act is done. As there is no additional problem solving content in the user's utterance (3), the system does not do anything in response.

The user continues with:

U: Could Celebrex cause headaches? (4)

As with (1), the IM asks the TM to interpret the utterance in the current problem solving state. Note that the IM must explicitly indicate that this interpretation should be relative to the context in which the system's interaction act (the *complete* of the *c-evaluate-action*) is done. This is just one example of the explicit synchronization necessary in order to implement the collaborative problem solving model in a distributed system.

The TM answers that the utterance (4) looks like an *initiate* of a *c-identify-situation* (as to whether celebrex causes headaches). With this choice of an interpretation, the Task Manager can update its model of what the user is doing. Meanwhile, the Behavioral Agent decides that the system should perform its part of the collaborative action in order to complete it. The TM does the *identify-situation* and responds that it is not the case that celebrex can cause headaches. The BA passes this answer to the GM, requesting that it inform the user in order to complete the collaborative act. This results in the following system utterance:

S: No, headaches are not an expected side-effect of celebrex. (5)

And again, the *inform* is done once the speech has been produced.

Meanwhile, the TM, in the process of updating its state based on the user having initiated the *c-identify-situation* regarding celebrex causing headache, has performed some plan recognition and thinks it is likely that the user may be trying to cure their headache. Note that up to this point, the user has not said anything about having a headache—this is purely an inference based on task and domain knowledge. The TM reports that this is a problem which should be resolved, although it leaves it to the BA to prioritize the system's objectives and decide what to do.

In this case, the BA decides to take initiative and requests that the TM suggest what to do. The TM responds that the system should initiate a *c-identify-situation* regarding whether the user has a headache. The BA sends this to the GM, resulting the following utterance:

S: Do you have a headache? (6)

Once the speech has been output, the GM announces that the *ask* is done. At this point both interaction acts ("headaches are not a side-effect of celebrex" and "do you have a headache") are awaiting grounding, and the question is awaiting a response from the user. When the user answers with:

U: Yes. (7)

Both system utterances (5) and (6) are grounded, so both pending interactions acts are marked as completed, and the system proceeds to interpret the user's utterance (7) in the resulting context.

The dialogue for another continues for another fifteen utterances as the system addresses the user's headache and then supports them with several other aspects of their medication regime. Unfortunately, space precludes an extended presentation.

7. RELATED WORK

As noted above, work has been done on general theoretical frameworks for collaborative interaction [3, 9, 11]. However, the focus of these models was more specifying the mental details (beliefs, intentions, etc.) of such collaboration, whereas the focus of our model is describing practical dialogues. Also, in these proposals, many details of what the models would look like are not given. The SharedPlans formalism [9, 11], for example, *does* expressly model the adoption of recipes (Select_Recipe, Select_Recipe_GR), but that is as far as it goes. We believe that our model will prove to be complementary to this formalism, with the remainder of problem solving acts either existing at some higher level (e.g. adopt/abandon/evaluate-objective), being added to the same recipe level (evaluate-recipe), or being part of the unspecified Elaborate_Individual/Elaborate_Group processes.

Our belief that human-machine interaction can occur most naturally when the machine understands and does problem solving in a similar way to humans is very close to the philosophy upon which the COLLAGEN project [16] is founded. COLLAGEN is built on the SharedPlan formalism and provides an artificial language, human-computer interface with a software agent. The agent collaborates with the human through both communication and observation of actions. COLLAGEN, as it works on a subset of the SharedPlans formalism, also does not explicitly model most of our problem solving acts.

Several dialogue systems have divided intention recognition into several different layers, although these layerings are at much different levels than our own. Ramshaw [14] analyzes intentions on three levels: domain, exploration, and discourse. Domain level actions are similar to our own domain level. The discourse level deals with communicative actions. The exploration level supports a limited amount of evaluations of actions and plans. These, however, cannot be directly used to actually build up a collaborative plan, as they are on a stack and must be popped before the domain plan is added to.

Lambert and Carberry [10] also had a three level model, consisting of domain, problem solving, and discourse levels. Their problem solving level was fairly underdeveloped, but consists of such recipes as *Build_Plan* and *Compare_Recipe_by_Feature* (which allow the comparison of two recipes on one of their features). The model does not include other of our problem solving acts, nor does it explicitly model collaboration, interaction acts, etc.

These models assumed a *master-slave collaboration paradigm*, where an agent must automatically accept any proposal from the other agent. Chu-Carroll and Carberry [2] extended the work of Lambert and Carberry, adding a level of proposal and acceptance, which overcame the master-slave problem. However, Chu-Carroll and Carberry (along with Ramshaw and Lamber and Carberry), assume a shared, previously-specified problem solving *plan* which is being executed by the agents in order to collaborate. This restricts collaboration to homogeneous agents which have identical problem solving plans, whereas in our model, there is no set problem solving plan, allowing agents with different individual problem solving strategies to collaborate.

Finally, Elzer [4] specifically mentions the need for a problem-solving model in discourse, citing dialogue segments similar to those that we give. However, she offers no proposal of a solution.

8. CONCLUSIONS

The collaborative problem solving model presented in this paper offers a concrete proposal for modeling collaboration between agents, including in particular between human and software agents. Our model is based on our experience building collaborative systems in several problem solving domains. It incorporates as many elements as possible from formal models of collaboration, but is also driven by the practical needs of an implemented system.

9. ACKNOWLEDGMENTS

This material is based upon work supported by Dept. of Education (GAANN) grant no. P200A000306; ONR research grant no. N00014-01-1-1015; DARPA research grant no. F30602-98-2-0133; NSF grant no. EIA-0080124; and a grant from the W. M. Keck Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the above-mentioned organizations.

10. REFERENCES

[1] J. Allen, G. Ferguson, and A. Stent. An architecture for more realistic conversational systems. In *Proceedings of IUI-2001*, Santa Fe, NM, January 14-17 2001.

[2] J. Chu-Carroll and S. Carberry. Conflict resolution in collaborative planning dialogues. *International Journal of Human-Computer Studies*, 53(6):969–1015, 2000.

[3] P. Cohen and H. Levesque. Intention is choice with commitment. *Artifical Intelligence*, 42:213–261, 1990.

[4] S. Elzer. The role of user preferences and problem-solving knowledge in plan recognition for expert consultation systems. In *Working Notes of the IJCAI-95 Workshop on The Next Generation of Plan Recognition Systems*, pages 37–41, Montreal, Canada, 1995.

[5] G. Ferguson, J. Allen, N. Blaylock, D. Byron, N. Chambers, M. Dzikovska, L. Galescu, X. Shen, R. Swier, and M. Swift. The Medication Advisor project: Preliminary report. Technical Report 776, CS Dept., U. Rochester, May 2002.

[6] G. Ferguson and J.F. Allen. TRIPS: An integrated intelligent problem-solving assistant. In *Proceedings of AAAI-98*, pages 567–573, Madison, WI, 28–30 July 1998.

[7] M.P. Georgeff. Actions, processes, and causality. In *Proceedings of the Workshop on Reasoning about Actions and Plans*, Los Altos, CA, 30 June–2 July 1986.

[8] B. Grosz and C. Sidner. Attention, intention, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[9] B.J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[10] L. Lambert and S. Carberry. A tripartite plan-based model of dialogue. In *Proceedings of ACL-91*, pages 47–54, Berkeley, CA, June 1991.

[11] K.E. Lochbaum. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4):525–572, 1998.

[12] J. McCarthy. First order theories of individual concepts and propositions. In J.E. Hayes, D. Michie, and L.I. Mikulich, editors, *Machine Intelligence*, volume 9, pages 129–174. Ellis Horwood, Chichester, England, 1979.

[13] M.E. Pollack. The uses of plans. *Artificial Intelligence*, 57, 1992.

[14] L.A. Ramshaw. A three-level model for plan exploration. In *Proceedings of ACL-91*, pages 39–46, Berkeley, CA, June 1991.

[15] A.S. Rao and M.P. Georgeff. An abstract architecture for rational agents. In *Proceedings of KR-92*, pages 439–449, Boston, MA, 25–29 October 1992.

[16] C. Rich and C.L. Sidner. COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8(3–4):315–350, 1998.

[17] D.R. Traum. *A Computational Theory of Grounding in Natural Language Conversation*. PhD thesis, CS Department, U. Rochester, December 1994.