Managing Communicative Intentions in Dialogue Using a Collaborative Problem-Solving Model

Nate Blaylock

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report 774

April 2002

Abstract

In order to understand natural language, it is necessary to understand the intentions behind it, (*i.e.*, *why* an utterance was spoken). We model dialogue as *collaboration* between agents. Communicative intentions can then be seen as how an agent is trying to affect the collaboration. Most previous work on intention-recognition approaches to dialogue has focused on only a small subset of agent collaboration paradigms (*i.e.*, master-slave), and thus is unable to account for dialogues in other paradigms, such as mixed-initiative collaboration. Previous work has also either modeled dialogues where the agents are only planning or dialogues where agents are only acting. This restricts dialogue-model coverage to only those cases and does not model dialogues where collaboration about acting *and* planning occurs.

In this paper, we present a collaborative problem-solving model of dialogue. This model is able to account for a much wider array of dialogues than previous models have covered. It covers the spectrum of collaboration paradigms (from master-slave to mixed-initiative) as well as dialogues where interleaved acting and planning are taking place.

We propose, for future research, to complete this model and to build a domain-independent intention-recognition system based on it for use within the TRIPS dialogue system.

This material is based upon work supported by Dept. of Education (GAANN) under Grant number P200A000306 and by the Office of Naval Research (ONR) under Grant number N00014-01-1-1015. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of above-mentioned organizations.

Contents

1 Introduction

Language does not occur in a vacuum. It is a means to an end. This work is done in the context of producing a *conversational agent*: a system which can interact with a human in natural language (conversational) as well as act and affect the world (*i.e.*, an agent). Figure 1 shows an abstract architecture for a conversational agent.

The mode of communication in this case is, of course, natural language. Upon hearing an utterance, the system must be able to *understand* what is meant by it. This goes much deeper than just understanding what the utterance means on the surface. The system must be able to understand much more. Why did the user utter what he did? What was he trying to accomplish by uttering it? We will call these conversational "ends" *communicative intentions* and the process of inferring them *intention recognition*.

Once the system understands what the user's intentions are, it must decide what to do in response to those intentions. The system's *behavior* determines what it does, either through *actions*, which affect the world, or by *communicating* with the user.

If the system desires to communicate with the user, it sends its intentions (*i.e.*, what it wants to accomplish by communication) to *generation*, which turns the intentions into *language* to be then uttered to the user.

In this work, we concern ourselves primarily with two questions:

1. What are communicative intentions?
2. How can they be recognized from language?

The first question, although assumed in much previous work, has never been explored directly in the literature. As we will discuss later, the second question is discussed by a large body of work; however, there is still a wide array of language in which intentions cannot be recognized by previous systems.

Another way of viewing this work, in terms of Figure 1, is that defining communicative intentions will give us the "language" to express the messages sent along the arrows labeled "communicative intentions." Determining how these intentions are recognized will give us the understanding sub-system, which maps language to communicative intentions.

The tasks of understanding and generation are typically divided into many levels of analysis (*i.e.*, phonology, morphology, syntax, semantics, pragmatics, *etc.*), at all of which an enormous body of research has been done. For this paper, we concern ourselves only with the topmost level, which we will call the *intention/language interface*, where intentions are converted to and from a high-level semantic form (*i.e.*, speech acts). Encoding communicative intentions into this form (which is beyond the scope of this work) is called *content planning*. Decoding a high-level semantic form into communicative intentions is called *intention recognition*.

For successful decoding to occur, there must exist some sort of encoding which is known to both the encoder and the decoder, and which can be used to unambiguously decode

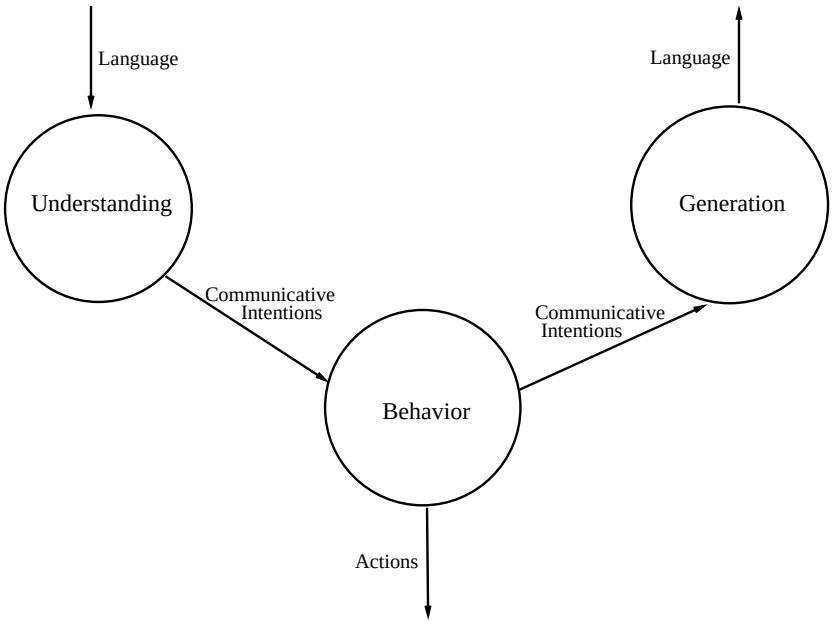

Figure 1: A Typical Conversational-Agent Architecture

the message. *Natural language discourse structure* describes how multiple utterances fit together, and serves as the encoding for the intention/language interface. *Collaboration* is intention-based communication between agents. Agents' individual actions are guided by *objectives*, and *problem solving* is the process agents use to try to accomplish their objectives.

This paper is a proposal for the following research.

1. Build a model of human collaborative problem solving and use it to define and describe communicative intentions.
2. Use that model to build a domain-independent intention-recognition system for use in a collaborative agent.

We will leave the areas of of reasoning with recognized intentions (behavior) and producing language from intentions (content planning) to future research. We do believe, however, that the use of our collaborative problem-solving model will greatly improve both the accuracy and the portability/domain independence of both.

In order to provide a foundation for the paper, we give a quick, bottom-up sketch of our collaborative problem-solving model. In Section 2 we describe previous work in this area. Following that we describe our model in depth, and then discuss preliminary results that we have obtained in using the model within our dialogue system. Finally, we propose a research plan and give some concluding remarks.

1.1 Terminology and Concepts

We begin here by defining a core set of concepts which we use in the description of our problem-solving model.

Objective: An objective is an abstract action that is driving the behavior of an agent. Objectives are defined by their effects and not specified how they are to be achieved. An objective may be something like "to open the door" or "to lift the piano."

Recipe: Our definition of recipe is similar to that put forth by Pollack ([Pollack, 1986a]). A recipe is a formula for achieving an objective starting from some situation. We allow for a very loose definition of recipe. All we require is that a recipe tell one what to do in the some situations in order to work towards achieving an objective. For other situations, a recipe may not be defined, in which case only by further refining the recipe (planning) or choosing a different recipe will the agent be able to work towards the objective. A recipe could be a simple list of actions to perform, or it could be a more complex learned behavior, which is valid over a large number of varying situations. Recipes may be previously known by an agent, or they may be constructed on the fly (through planning).

Atomic Action: Directly executable actions in the domain.

Resource: A resource is an object utilized for the execution of an action. It may be something consumed during execution (*e.g.*, gasoline in an engine), or just used during the time of execution (*e.g.*, a truck).

Situation: The state of the world (or a possible world). Takes after the situation calculus ([McCarthy, 1979]). An agent's beliefs about situations may be incomplete or incorrect.

With these concepts now defined, we now describe a model of individual-agent problem solving.

1.2 The Problem-Solving Level

The Problem-Solving Level describes the individual reasoning and behavior of an agent. This is based on beliefs and commitments to objectives and recipes.[1] When an agent is acting alone, it does not need to communicate any intentions. However, this basic model of single-agent problem solving is useful and helps us to describe multi-agent collaborative problem solving, which does involve communication.

There are three basic phases of problem solving that an agent goes through. In practice, some of these phases may be skipped, revisited, *etc.*

[1] We are not trying to compete with formal models of agent behavior (*e.g.*, [Rao and Georgeff, 1991; Cohen and Levesque, 1990]). However, for modeling collaboration, it is useful to have an abstraction for modeling other agents' thought processes.

1. *Determining Objectives:* An agent determines which objectives it should adopt (intend), which it should focus on, and which it should no longer have.

2. *Determining and Instantiating Recipes for Objectives:* For each objective it has, an agent decides on a recipe it will use to achieve it. A recipe can be chosen from a pre-stored recipe library, or it can be created on the fly by planning. An agent may follow a certain recipe for a while and then decide to discard it and adopt a different recipe for that objective.

3. *Acting:* To act, an agent follows a recipe. If the next step of the recipe is a complex action (*i.e.*, an objective), the reasoning process starts over, and the agent must determine whether or not to adopt that objective. If the next step is an atomic action, an agent can evaluate whether or not it wants to perform it, and then perform it if it so decides.

It is important to note that this model does not dictate exactly how an agent *should* perform problem solving. This framework allows for a wide variety of *problem-solving strategies* an agent might employ. An agent which only follows recipes and reasons little about its objectives and recipes will be reactive but brittle, whereas an agent that constantly re-evaluates its objectives will be flexible, but slow. This reflects the variance of problem-solving strategies in humans: some of us are cautious and slower to react, while others make decisions only once and then stick to them. In the human world, people with problem-solving strategies from one extreme can still collaborate with people from the other extreme. Our model needs to allow for collaboration among such diverse agents as well.

1.3 The Collaborative Level

The Collaborative Level describes how agents solve problems *together*, coordinating their action through communication. One can actually just think of this as a multi-agent extension of our single-agent problem-solving model. Collaborative agents determine which collaborative objectives they will adopt, which they will put into focus, and which they will abandon. They must determine which recipes they will use for objectives (possibly creating recipes together). They must also be able to follow recipes together, and where necessary, perform joint actions.

This level is actually more complicated than just relabeling the single-agent problem-solving phases as "collaborative." As opposed to a single agent, which has beliefs and commitments stored in a single place (*i.e.*, its mind), collaborative problem-solving state only emerges as the result of individual beliefs and commitments. This means that a single agent cannot directly affect the collaborative problem-solving state (*i.e.*, which objectives are adopted, *etc.*) Instead it can only *attempt* to change the collaborative state through communication.[2]

[2]We do not treat the issue of teamwork without communication here. For a review of that work see [Van Wie, 2001].

1.4 The Interaction Level

The Interaction Level describes the *individual* actions that agents can perform to attempt to change the collaborative problem-solving state. One agent *initiates* a proposal, which the other agent can *continue*, by refining it, or it can *complete* the proposal by accepting it and changing its own internal state in the necessary ways. An agent can also *reject* another agent's proposal, which causes the proposal to fail and the collaborative problem-solving state to remain unchanged. The interaction level also describes the grounding that must take place to provide the necessary mutual belief for collaborative-level problem solving. These interaction acts are what we use to represent communicative intentions.

With this model as a backdrop, let us revisit part of the communication process we described above. When an agent has a (private) intention to somehow affect the collaborative problem-solving state, say to adopt a new collaborative objective, it must formulate an interaction-level action (say, to initiate the adoption), which it will then communicate. In order to communicate, it must first encode this intention into language. This is the content planning process.

The agent receiving the communication must take the language signal and try to decode it by determining the original intentions the sending agent had (*i.e.*, initiating the adoption of a new collaborative objective). This is the intention-recognition process.

Intention recognition is possible only when the encoding agent conforms to certain shared conventions, which allow for easy and (hopefully) unambiguous decoding of the intentions. These shared conventions, of which this collaborative problem-solving model is a part, constitute the discourse structure.

Given this foundation, we now explore previous work related to our own.

2 Previous Work

2.1 Collaborative Problem-Solving Models

In this section we describe previous work in models of collaborative problem-solving among agents. We first describe work done on joint activity, and then on agent interaction. We then describe the COLLAGEN system, which integrates a model of joint activity with an agent interaction model. Finally, we discuss some shortcomings in this area.

Joint Activity

Joint activity describes collaboration and action among agents. While much work has been done on formalizing joint activity and intention ([Searle, 1990; Levesque *et al.*, 1990; Grosz and Kraus, 1996; Grosz and Kraus, 1999]), the focus of this work has mostly been on formalization (an area in which our model is currently weak) and not on collaboration.

We describe here several models of collaboration between agents: first, the SharedPlan theory and then, other models from the agents literature.

SharedPlans The SharedPlan formalism ([Grosz and Kraus, 1996; Grosz and Kraus, 1999]) was created in part to explain the intentional structure of discourse ([Grosz and Sidner, 1986; Grosz and Sidner, 1990; Lochbaum *et al.*, 2000]). It describes how agents collaborate together to form a joint plan. The SharedPlan model has four operators which are used by agents in creating a SharedPlan.

- *Select_Rec*: At an individual level, given a subgoal, selects a recipe to be used to attain it.
- *Elaborate_Individual*: At an individual-agent level, takes a recipe and decomposes it, filling in all the necessary details.
- *Select_Rec_GR*: Intuitively, the same as Select_Recipe, only at the multi-agent level.[3] A group of agents select a recipe for a subgoal.
- *Elaborate_Group*: The multi-agent equivalent of Elaborate_Individual.

Using these four operators, a group of agents collaborates until it has completely specified a *full SharedPlan*.

Lochbaum ([Lochbaum, 1991; Lochbaum, 1994; Lochbaum, 1998]) developed an intention-recognition algorithm based on the SharedPlan formalism and modeling the process of *plan augmentation*. Upon hearing an utterance, an agent ascribes certain intentions and beliefs to the speaker. If it is willing, the agent also adopts those intentions and beliefs. As a result of the new beliefs and intentions, the SharedPlan is augmented.

Other Models Apart from work in discourse, some work on collaborative agents has been done in the field of Distributed AI, although this work is at a very high level. The work in [Burmeister and Sundermeyer, 1992], for example, describes a BDI agent and the only level of collaboration defined is that the agent can communicate. (It says nothing of *how* agent communicate.) In [Osawa and Tokoro, 1992], collaboration consists of each agent proposing an individual plan and then individual agents gleaning information from the proposed individual plans to make a better plan (which they couldn't have done based solely on its own knowledge). There is no collaboration (as we define it) in this model and homogeneous agents are assumed.

Some work closer to ours was done by Wooldridge and Jennings ([Wooldridge and Jennings, 1996; Wooldridge and Jennings, 1994; Wooldridge and Jennings, 1999]). They formalize a model of collaboration that happens in four phases.

1. *Recognition:* An agent recognizes the potential for collaboration.
2. *Team Formation:* An agent forms a team to collaborate with.
3. *Plan Formation:* The team comes up with a plan.

[3]Individual and group operators entail different constraints on individual intentions and beliefs. However, this is not important for understanding the formalism as a collaborative problem-solving model.

4. *Team Action:* The team executes the plan.

Following these stages in order, an agent puts together a team, the team forms a plan and then executes it.

Agent Interaction

We define agent interaction to be models of how agents interact and *negotiate* their goals, plans, *etc.* when working collaboratively. In general, much work has been done on autonomous agent negotiation for everything from eCommerce auctions to agent service brokering, using mostly game theory or argumentation theory ([Jennings *et al.*, 2001]). We discuss here only the work done by Sidner ([Sidner, 1994a; Sidner, 1994b]), which looks at the domain of discourse.

Sidner defines an artificial language for interaction and uses it to analyze several interaction dialogues. The model allows for things such as proposals, counterproposals, acceptance, rejection, as well as for grounding acknowledgments. As a simplification, the model is defined for an artificial language, which explicitly marks all actions (including grounding). This eliminates the need for intention recognition. It is not clear whether this model will be useful in a domain (such as natural language) which requires intention recognition.

We are currently looking at the differences between Sidner's model and our own (see Section 3.4) to see if we can benefit from the insights of her model. This is one area of proposed future research (see Section 5).

The COLLAGEN System

The COLLAGEN system ([Rich and Sidner, 1997; Rich and Sidner, 1998; Rich *et al.*, 2001]) is an intelligent user-interface system which models human-computer interaction on principles of discourse. It is a specific instance of Lochbaum's algorithm and works on a subset of the SharedPlan formalism.

In COLLAGEN, communication between the user and system is simplified. Based on the current discourse state, the system presents the user with a list of possible contributions to the dialogue. The user chooses one which becomes his "utterance." The system also observes user activities in the underlying program.

Based on these observations, (both linguistic and non-linguistic) the system keeps a discourse state which it uses to understand user utterances and activities. Because the system understands the context of the interaction, the user can interact with it more naturally.

Shortcomings of Previous Collaborative Problem-Solving Models

The above-mentioned research projects have mostly had different research goals from ours and, as a result, has some shortcomings when it comes to applying them to modeling a wide range of dialogue. Work in the agents field, for example, has typically dealt with a group of (all non-human) homogeneous agents which do not need to conform to human

problem-solving strategies. Because agents in our work collaborate with humans, they need to be able to collaborate *as humans do*.

The main focus of the SharedPlan model has been to formalize agent intentions and beliefs in forming and sharing joint plans, something which is still weak in our model. However, for our purposes — modeling a wide array of collaborative dialogues — there are several shortcomings in this model which we discuss for the rest of this section.

First, SharedPlans only models collaboration for planning between agents. It does not model the collaboration that occurs when agents are trying to *execute* a joint plan (although it does specify the needed intentions and beliefs for agents executing joint plans). In the domains we are working in, the user and system collaborate in both planning and acting. This sometimes happens in a serial fashion (the agents formulate a joint plan and then execute it), or it can be interleaved (the agents begin to execute a partial plan and plan "as they go.") We need to be able to model collaboration about (possibly interleaved) planning and acting.

Second, the SharedPlans formalism models the formulation of joint plans with the four operators previously discussed: *Select_Rec*, *Elaborate_Individual*, *Select_Rec_GR*, and *Elaborate_Group*. Although these operators were sufficient to allow the formalization of group intentions and beliefs about joint plans, they do not provide enough detail for us to model collaborative dialogue at an utterance level (*i.e.* the level at which agents communicate).

As an example, consider the *Elaborate_Group* operator. This has the function of decomposing a recipe, instantiating the parameters (including which agent or subgroup will perform which action at what time and which resources will be used), and making sure the rest of the group has similar intentions and beliefs about the plan. Any one of these activities could be collaborated about over many utterance exchanges among agents. For our work on communicative intentions, we need to be able to model the intentions behind a single utterance.

Grosz and Kraus leave the details of the SharedPlan operators to be implemented within an individual agent ([Grosz and Kraus, 1996]). However, this would restrict agents to collaborating only with agents with a homogeneous definition of the operators. Agents which did not perform *Elaborate_Group* the same way, for example, would not be able to collaborate.

A more general solution is to include the general details of these operators into the model. Sidner (see above), for example, shows how agents could negotiate about augmenting SharedPlans.

We believe that our model is compatible with SharedPlans and specifies the general details of the SharedPlan operators. (See Section 3.3 below.)

2.2 Intention Recognition

Recognizing the intention behind an utterance can be seen as a special case of *plan recognition* — inferring an agent's plan based on observed action. Much previous work in intention recognition has taken a plan-recognition approach based on speech act theory. We first briefly describe speech act theory, which connects language to action. We then discuss previous intention-recognition research and then summarize its shortcomings.

Speech Acts

Austin ([Austin, 1962]), Grice ([Grice, 1957; Grice, 1969; Grice, 1975]), and Searle ([Searle, 1975]) all noted that human utterances can actually cause *changes* in the world. The utterance "I pronounce you man and wife," said by the right person in the right context actually causes two people to be married. More subtly, the utterance of "John is in the kitchen" may have effect of causing the hearer to believe that John is in the kitchen.

Utterances can have preconditions and effects, the same as other non-linguistic actions. We can also build plans that contain utterances as well as other actions. A full discussion of speech acts is beyond the scope of this paper. It is important to realize, however, that treating utterances like actions (speech acts) allows us to use plan recognition within dialogue systems. Take for example, a speaker who has an intention to cause us to believe that John is in the kitchen. His plan is to utter to us the words "John is in the kitchen" to us. We, the hearer, then must be able to recognize the speaker's original intention (whether to inform us that John is in the kitchen, to get us to go into the kitchen to see John, to hint that we should *not* go into the kitchen because John is in there, *etc.*) Different planning contexts will result in different interpretations for the same utterance.

Allen, Cohen, and Perrault

Allen, Cohen, and Perrault were the first to computationalize a theory of speech acts. They showed that in question answering systems, users expected the system to recognize their unstated goals in order to provide more helpful responses to questions ([Cohen *et al.*, 1982]).

Cohen ([Cohen, 1978; Cohen and Perrault, 1979]) concentrated on using plan synthesis together with speech acts for content planning. Allen ([Allen, 1979; Allen and Perrault, 1980; Allen, 1983]), on the other hand, used plan recognition and speech act theory for intention recognition. We concentrate here only on Allen's work.

Allen studied transcripts of actual interactions at an information booth in a Toronto train station. A typical exchange was something like this ([Allen, 1983]):

Patron:	When does the Montreal train leave?
Clerk:	3:15 at gate 7.

Note that although the patron only requested the departure time, the clerk also volunteered information about the departure gate as well. Presumably, the clerk *recognized* the plan of the patron (to board the train), and realized that the patron would also need to know where the train departed and volunteered that information as well. Allen called this behavior *obstacle detection*.

Allen's system took the direct speech act of the utterance, and, using certain inference rules and heuristics to apply them, performed backward chaining in order to infer the user's plan. Heuristics included things such as, if a person wants P, and P is a precondition of action ACT, then the person may want to perform ACT; or if a person wants to know if P is true, they may want P to be true (or false).

Using these inference rules, the system was able to recognize not only indirect speech acts, but also the user's domain plan. However, Allen's system only worked for one-utterance dialogues.

Multiple-Utterance Intention Recognition

Sidner and Israel ([Sidner and Israel, 1981]) extended Allen's work to multiple-utterance dialogues. Their work took previous discourse context into account in interpreting a new utterance. Each utterance caused the system to update its beliefs about the user's beliefs, wants, goals, and plans, based on Grice's theoretical work ([Grice, 1957; Grice, 1969]) as well as plan recognition similar to Allen's (see above).

Carberry ([Carberry, 1987; Carberry, 1990b]) also extended Allen's work to account for multiple utterances. She used a plan decomposition hierarchy similar to Kautz' (see Section 2.3) which held information about decomposition of plans in the domain. Her system explicitly kept track of this hierarchy and filled it in as the user navigated and expanded different nodes. This type of plan-recognition system accounted for both bottom-up (talk about actions first) and top-down (talk about goals first) dialogue.

Carberry's system did intention recognition in a two-phase manner. The first phase performed *local analysis* on input, which tried to identify the speaker's *immediate* goal, regardless of context. Afterwards, one of these goals was chosen based on *global analysis* (the second phase), which tried to fit one of these goals into the context of previous utterances.

Focus was an important part of this system. Carberry found that speakers usually navigate and fill in the plan tree in a predictable way. Thus, if focus is on a certain node, the system can calculate the most likely shifts of focus from that node. This allowed the system to use context as well as expected tree navigation conventions (*i.e.*, discourse structure) to filter among the possible immediate goals from local analysis and place them into the context.

Although Carberry's work was able to account for dialogues describing complex plans, it only worked for utterances that talked directly about actions and goals in a domain. Other dialogue phenomena (such as correction and clarification subdialogues) were not addressed.

Dialogue and Domain Plans

Litman and Allen ([Litman, 1985; Litman, 1986; Litman and Allen, 1987; Litman and Allen, 1990]) extended Carberry's earlier work to better account for various dialogue phenomena. Although a dialogue's focus is on the domain, there seem to be several meta-layers which help ensure robust communication.

Essentially, Litman and Allen added a new layer to the plan-recognition system, that of meta-level[4] plans, which operate on other plans. Previous intention-recognition systems had only accounted for domain-level plans (as well as speech act level plans). Litman and Allen's system was able to account for a number of dialogue phenomena, including that of

[4]Litman and Allen actually called these *discourse* plans. In light of subsequent work, however, these are better characterized as meta-level plans.

clarification subdialogues. For example, consider the following dialogue ([Litman and Allen, 1990]):

teacher:	OK the next thing you do is add one egg to the blender, to the shrimp in the blender.
student:	The whole egg?
teacher:	Yeah, the whole egg. Not the shells.
student:	Gotcha. Done.

The domain-level plan here one of cooking. The student's first utterance ("The whole egg?"), however is uttered because of confusion about the previous utterance. Instead of replying directly to the teacher's instruction, the student asks a clarification question, which Litman and Allen call an IDENTIFY-PARAMETER meta-plan (*i.e.*, a plan to clarify the parameter — the egg — to be used in the domain-level plan). The teacher responds to this question and then the student "pops" back down to the domain-level plan again and says "Done."

Litman and Allen allowed this stack-like structure of dialogues and recognized not only domain-level plans, but also the domain-independent meta-level plans. Other examples of meta-level plans included: CORRECT-PLAN (changes a plan when unexpected events happen at runtime), INTRODUCE-PLAN (shifts focus to a new plan), and MODIFY-PLAN (changes some part of a plan).

The addition of meta-level plans allowed fuller coverage of various dialogue phenomena, especially correction and clarification subdialogues. While we see this work as an early foundation for our own, we note that the meta-level plan library was somewhat ad hoc and quite underdeveloped, dealing mostly with information-seeking dialogues and not the actual creation and discussion of plans (see Section 2.2).

Other Plan Levels in Dialogue

There have been several efforts to extend the work of Litman and Allen, further categorizing the types and levels of plans that can exist in dialogue. We first discuss the work of Lambert, and then that of Ramshaw. We then look at further refinements by Carberry *et al.*, Ardissono *et al.*, and finally Chu-Carroll and Carberry.

Lambert Lambert ([Lambert and Carberry, 1991; Lambert, 1993]) proposed a three-level model of dialogue, consisting of the domain and meta levels of Litman and Allen as well as a level of discourse plans, which specifically handled recognition of multi-utterance speech acts. The follow utterances, for example, constitute a warning (a discourse plan) only when taken together ([Lambert and Carberry, 1991]).

U1:	The city of xxx is considering filing for bankruptcy.
U2:	One of your mutual funds owns xxx bonds.

The separation of discourse plans allows the recognition of speech-act-like phenomena such as warnings and surprise at a multi-utterance level.

The focus of this work was on the discourse level; no new work was done at the meta-level.

Ramshaw At the same time as Lambert, Ramshaw ([Ramshaw, 1989b; Ramshaw, 1989a; Ramshaw, 1991]) proposed a differently separated three-level model. Instead of a discourse level, Ramshaw proposed an *exploration* level. The intuition is that some utterances are made simply in the attempt to *explore* a possible course of action, whereas others are explicitly made to attempt to *execute* a plan.

Ramshaw's model allows the system to distinguish between the two cases, He uses a stack-based approach, with exploration-level plans pushed on lower-level plans. Unfortunately, this approach means that exploration-level and domain-level plans must be separate. Once one starts exploring, one cannot talk about the plan-execution level until done exploring. As observed in [Carberry *et al.*, 1992], this prevents understanding of contingent commitments when one is at the exploration level. One cannot build two complex, competing plans and compare them. Also, the model could not handle bottom-up dialogues (dialogues which start with atomic actions and build their way up).

Although Ramshaw's model allows for some problem-solving behavior, including comparisons of different plans, it does not model this collaboratively. Instead, individual comparisons can only be detected, but they do not affect the planning state.

Carberry *et al.* The research in [Carberry *et al.*, 1992] incorporated Ramshaw's work into the previous work by Lambert (see above). They overcame the problem with contingent commitments and also modeling the building and comparing of complex plans. It also allowed for the handling of both top-down (goal first) and bottom-up (actions first) dialogues.

However, the problem-solving level was fairly undeveloped, mentioning only exploring different recipes, choosing the best, and doing the same at lower levels. There are several shortcomings of this approach.

First, the plan structure requires that these meta plans be done linearly. In other words, one must first explore all recipes, and once this is fully done, choose the best, *etc.* It does not seem to support the revisiting of previous decisions by an agent

Second, there is no execution in the model. It only provides for dialogues about planning. Also, although there is a limited problem-solving model, there is no notion of collaboration. It is not clear how or if each agent could contribute separately to the plan being built. Sample dialogues are all of the master-slave type ([Grosz and Sidner, 1990]), where one agent is proactive in planning and the other basically serves as an information source. It is unclear if this model would be suitable for domains outside of information-seeking dialogues.

Ardissono *et al.* The work in [Ardissono *et al.*, 1996] modified the work of Carberry *et al.* by generalizing the execution model. Instead of Domain, Problem-solving, Discourse, the model became Goal, Problem-solving, Action.

The problem-solving level consists of just two high-level plans: *Satisfy* and *Try-execute*. These decompose into an execution model for an agent. In order for an agent to *Try-execute* an action, it first checks constraints and *Satisfies* them if necessary; it then verifies preconditions, does the action and checks the results. Utterances are explained based on some step in this execution model. For example, consider the following dialogue ([Ardissono *et al.*, 1996]).

Mark: Sell me a bottle of whiskey.
Lucy: Are you over 18?

Here, Lucy's response can be seen as her doing a *Try-execute* of selling the whiskey, needing to know if Mark is over 18 in order to verify a constraint.

The domain level is the same as other previous work. The action level, although never really explained, is presumably the actual execution of actions and is not really a model of a separate level of intentions interpreted from an utterance.

Although this model describes an execution loop in an agent well, it cannot account for most discourse phenomena that earlier models were able to account for. One major shortcoming is that the model does not take into account agent planning. It assumes that an agent is executing a predetermined plan. This precludes most of Litman's meta-level plans (introducing a new plan, identifying parameters, correcting a plan, *etc.*) and Ramshaw's plan comparison, as well as Lambert's discourse-level plans. In fact, it does not seem likely that this model of intention recognition would be applicable outside a domain where an agent is executing predetermined plans.

Chu-Carroll and Carberry

Chu-Carroll and Carberry ([Chu-Carroll and Carberry, 1994; Chu-Carroll and Carberry, 1995; Chu-Carroll and Carberry, 1996; Chu-Carroll and Carberry, 2000]) extended the three-level model of Lambert and added a fourth level, belief. They also changed the model so that it distinguished between proposed and accepted plans and beliefs. This greatly extended the previous models to include *negotiation dialogues*, where agents have conflicting views and collaborate to resolve them.

Shortcomings of Previous Work on Intention Recognition

All of these multi-level intention-recognition systems share several shortcomings. First, they are based on a static plan library. At the domain level, this prevents the system from recognizing novel user plans. It also precludes the system and user from building novel plans together based on separate domain planning knowledge.

Because problem-solving-level plans are also static, the system can only interact with the user based on the limited number of *problem-solving strategies* stored in its plan library. For example, a problem-solving plan in ([Carberry *et al.*, 1992]) expects the user and system to collaborate by first exploring the best recipes for a goal, then choosing the best specialization

of a recipe, and finally building the plan by decomposing and instantiating the recipe. If the user instead wanted to choose a certain recipe, attempt to instantiate it and then come back and explore another instead, he would be unable to if that specific problem-solving plan was not in the system's plan library.

In addition to these, there are two main shortcomings which motivate our work. We first discuss the need for multiple collaboration paradigms. We then discuss collaboration about planning and acting.

Collaboration Paradigms A *collaboration paradigm* describes the respective roles and authority each agent has during collaboration. Agents may, for example, have different social statuses, giving rise to different collaboration paradigms. If two agents are on equal social footing, then they may both be free to make and reject proposals whenever they feel like it. This is a *mixed-initiative* collaboration paradigm ([Chu-Caroll and Brown, 1997]). At the other extreme is the *master-slave* collaboration paradigm ([Grosz and Sidner, 1990]), in which one agent completely controls the flow of the collaboration as well as the decisions made. There is a whole spectrum of collaboration paradigms between the extremes of mixed-initiative and master-slave. In a *boss-worker* paradigm, for example, the socially higher (boss) agent likely controls most of the discussion, but the worker agent may be expected to make contributions at certain levels but not be allowed to disagree with the boss.

With the exception of ([Chu-Carroll and Carberry, 2000]), previous work in intention recognition has only modeled master-slave collaboration. Most previous research was restricted to information-seeking dialogues (*e.g.*, [Carberry, 1990b]), where collaboration consists of the user (the master) getting information from the system (the slave). Although the system may possibly ask clarification questions, it cannot take task-level initiative ([Chu-Caroll and Brown, 1997]) and is not party to the user's planning decisions. Expert-apprentice dialogues (*e.g.*, [Grosz, 1981]) also fit this model. In these, the expert is the master, and the apprentice only follows the plan the expert puts forth.

This master-slave assumption limits the types of collaboration which can be modeled. We are interested in modeling the entire spectrum of collaboration paradigms, from master-slave to mixed-initiative.

Collaboration about Planning and Acting The second main shortcoming of these models is the same as that we discussed above about SharedPlans: modeling collaboration about planning and acting. Previous intention-recognition work has had the implicit assumption that either only planning was taking place in the dialogue (*e.g.*, [Ramshaw, 1991; Chu-Carroll and Carberry, 2000]) — with the plan to be executed at some later time — or that only execution of some previously created plan was occurring (*e.g.*, [Ardissono *et al.*, 1996]).

As we discussed in Section 2.1, in certain domains, agents collaborate about acting and planning, often interleaving the two. We need a model of collaboration which covers all of these cases.

2.3 Plan Recognition

In this section,[5] we discuss previous work in the field of plan recognition. As stated above, intention recognition is a special case of plan recognition and most intention-recognition implementations are based on work in plan recognition. We first present two early pieces of research which are considered to be the foundations of plan recognition. Although there were many other early general plan-recognition systems, almost all subsequent work on plan recognition has roots in these two systems. We first discuss the BELIEVER system and then Kautz' work on generalized plan recognition. We then discuss several shortcomings with prototypical plan recognition and attempts to overcome them.

BELIEVER

One of the first plan-recognition projects is in [Schmidt *et al.*, 1978], which showed that humans do indeed use plan recognition and then built a plan recognizer. In building BELIEVER, Schmidt *et al.* conducted psychological experiments to see how humans do plan recognition. They then built the BELIEVER system using these results. We will first describe their experimental results and then the workings of their system.

The Experiment In the experiments, subjects were given descriptions of actions in sequence. At certain times, the subject would be asked to summarize what he thought the actor was trying to do.[6]

In their summarizations, the subjects would attribute beliefs and intentions to the actor, particularly about his plans and goals. The subjects were doing plan recognition based on the observed actions.

One interesting observations made by the experimenters was that the subjects reported their beliefs about the plans of a user as a single hypothesis, even when much ambiguity existed. When details were not available, subjects would give a "sketchy hypothesis" for the plan, such as "John is getting *something* to eat." When further observations invalidated this single hypothesis, a new hypothesis would be proposed which fit the new data.

In other words, these experiments showed that humans make the best guess they can as to the plans of the actor they are observing, instead of keeping a disjunctive list of possible plans. As new data arrives, humans revise this hypothesis as necessary. As Lesh points out ([Lesh, 1998]), despite these findings, it is interesting that most plan-recognition systems since have done just the opposite, generating all possible hypotheses, and then trimming them down as more data comes along.

The System The BELIEVER system was built to try to mimic the human plan-recognition behavior found in the experiments described above. BELIEVER was given a priori knowledge of the world and about specific plans. It would then be given an observation, from which it would build a plan hypothesis for the actor.

[5] This section is based on [Blaylock, 2001].

[6] Other experiments also asked subjects to recall the events or to predict what the next action would be. This particular set of experiments, however, is the most relevant to our discussion.

BELIEVER matched incoming observations into a pre-built *expected plan structure*. The expected plan structure was a graph of actions which connected different enables/in-order-to relations among the preconditions and results of the different actions. A set of *propositions* existed which asserted the relationship between actions. A violation of a proposition would result in that hypothesis being thrown out. Matched observations would be put into the *grounded plan structure*. Once all actions had been observed/grounded, the plan was considered fully grounded.

When a hypothesis was deemed incorrect, (upon receiving a conflicting observation) the hypothesis revision process would run. Depending on the class of revision (*i.e.*, how the hypothesis had been deemed unacceptable), different rules would apply, which would generate a set of possible revisions. These revisions were then checked and one selected. This checking/selection algorithm, however, was not implemented in BELIEVER and was never given in any detail.

Probably the biggest contribution of BELIEVER was the experiments that showed that humans use plan recognition. This observation was among the beginnings of the plan-recognition field. The implementation, however, was never fully implemented nor specified.

Kautz' Generalized Plan Recognition

The other foundational work in plan recognition was that of Kautz ([Kautz and Allen, 1986; Kautz, 1987; Kautz, 1990; Kautz, 1991]). Kautz cast plan recognition as the logical nonmonotonic inference process of circumscription. This logical cast on plan recognition allowed him to use the a very rich knowledge representation (essentially that of first order logic).

Kautz represented the space of possible plans as an event hierarchy, which included both abstraction and decomposition (subaction) relations. Certain actions were labeled as *end* actions, meaning that they were an end unto themselves, a possible ultimate goal of an actor.

By making a few assumptions (such that this event hierarchy was complete), plan recognition became a problem of circumscription. Observations (also represented in first order logic), were used to infer plans nonmonotonically. Minimalization was then used to further reduce the number of hypotheses.

The biggest problem for this generalized plan-recognition system (which was the base for almost *all* subsequent plan-recognition systems) is that it is not tractable for large plan libraries and/or large numbers of observations. Kautz calculated that the complexity of the algorithm he gave was $O(2^n|H_E|^n)$, where n is the number of observations and $|H_E|$ is the size of the plan library. Also, because it is based on logical deduction, the model fails when there is a lot of ambiguity. An ambiguous observation (*i.e.*, an observation which could explain more than one plan) simply creates a disjunction of possible plans. There is no method for choosing just one.

Shortcomings of Previous Plan-Recognition Systems

In this section, we discuss several shortcomings of prototypical plan-recognition systems, some of which we alluded to above. The most ominous problem to any plan-recognition system is speed. In larger domains, ambiguity also becomes a problem that needs to be dealt with. Finally, a plan-recognition system needs to be able to deal with any mistaken inferences it makes, in light of new contradictory evidence.

Runtime Lesh ([Lesh and Etzioni, 1996]) points out that previous plan-recognition work has typically only been investigated in domains where the number of goals were less than one hundred. This brings up the question of whether Kautzian plan-recognition techniques can be scaled up to more realistic domains with thousands or even hundreds of thousands of goals and plans. Due to the exponential runtime of most plan-recognition systems, this is very doubtful. In fact, even in small domains, plan recognition is still a very costly subroutine.

Although we consider this problem still to be very much an open issue, there have been a few attempts to try to confront it. Mayfield ([Mayfield, 1992]) suggests a very pragmatic approach. His plan-recognition system uses a utility function to determine if it should continue to search for a solution. Thus his system weighs the tradeoff between accuracy and speed. Because of the upward-chaining nature of typical plan recognition, a system finds intermediate goals before finding an "ultimate goal" (and *end* in Kautzian terms). These partial results may be useful enough to a process which uses the plan-recognition results.

Of course, as Mayfield points out, utility can only be measured in an environment where plan recognition is being used within a larger system. His plan recognizer would do the best it could within a reasonable amount of time and then send the results to the next phase of processing in the larger system. This component (which used the results of plan recognition) would then know if the results it received from the plan recognizer were good enough or not. If a higher-level goal, or more accuracy were needed, the component would requery the plan recognizer, which would continue its search until the time spent outweighed utility. The process continued in like manner until the consuming component was satisfied.

Although this is not a panacea for the scalability problem, Mayfield's utility approach is definitely a step in the right direction.

Lesh ([Lesh and Etzioni, 1995; Lesh and Etzioni, 1996; Lesh, 1997; Lesh, 1998]) has taken plan-recognition research to the next level of scalability. Using automatic plan library construction techniques (see below), he has constructed plan libraries in several domains which contain up to 500,000 goals.

Lesh's research has been on *goal recognition*, which is a special case of plan recognition where only the user's goal is recognized, not the plan by which the user is trying to accomplish the goal. Lesh points out that goal recognition has several potential applications, including that of intelligent interfaces. We also believe goal recognition can be used to improve plan recognition. A fast goal recognizer could be used to sufficiently narrow the search space to allow a slower plan recognizer to be applied afterwards.

Lesh's system recognizes goals in time *logarithmic* to the number of goals in the plan library. It does this by using biases to prune away sections of incompatible goals.

We applaud Lesh's efforts to scale up plan-recognition systems. However, there is still much work to be done in this area. Lesh's algorithm only works in such fast time because his set of goals are represented in a subsumption hierarchy. Many of the 500,000 goals in his system are actually conjunctions of more primitive goals. This subsumption allows his algorithm to prune away huge sections of the search space at a time. It is not clear to what extent dialogue-type goals, which are not typically conjunctions, would benefit from this pruning.

Ambiguity Increasing the number of possible domain plans and goals grow typically also increases the level of ambiguity, *i.e.*, where more than one plan *could* describe the data. Two approaches to this problem have been taken. The first, somewhat unique approach to natural language dialogue, is the use of clarification, *i.e.*, querying the user to help resolve the ambiguity. The second is to use uncertainty to allow the system to reason with the likelihood of any given interpretation. We discuss these here.

- **Solution: Clarification Dialogue when Necessary**

  The most straightforward solution to the problem of ambiguity, which humans have been shown to employ, is to just "wait-and-see" ([Schmidt *et al.*, 1978]). If there is no need to disambiguate at the present state, then there is no need to worry. Humans often infer as much as is "reasonable" and when too much ambiguity is met, they just wait for more input to help them disambiguate ([Schmidt *et al.*, 1978]).

  Van Beek and Cohen ([van Beek and Cohen, 1991]) suggest that, in dialogue systems, it is not always even necessary to resolve ambiguity to give a proper response. Oftentimes, the response would have been similar regardless which plan (among the ambiguities) the user has. Alternatively, the system can generate an "ambiguous"reply which covers all the ambiguous cases.

  When ambiguity *does* matter, Van Beek and Cohen suggest ([van Beek and Cohen, 1991; Cohen *et al.*, 1994]) a method which humans often use to resolve ambiguity, that of asking for clarification. Clarification dialogue simply (directly or indirectly) solicits information from the user for disambiguation. In other words, if an ambiguity exists, one simply needs to ask. This is a method employed in COLLAGEN ([Lesh *et al.*, 1999]).

  One downside of the clarification approach, however, is that, if done too often, it can quickly throw a dialogue into disarray. Also, clarification is usually more effective if there are only a few ambiguities to choose among. Clearly, if clarification is to be used, it must be used sparingly. Other methods are also needed.

- **Solution: Using Uncertain Inference**

  The second approach to the problem of ambiguity is to use uncertain inference. This at gives each possible solution some measure of likelihood which the system to more easily choose among them. While this does not "solve" the ambiguity problem per

se, it does give the system a lot more leverage for dealing with it intelligently. We discuss here some of the many approaches which have been used.

Dempster-Shafer

Carberry ([Carberry, 1990a]) showed how to incorporate Dempster-Shafer probabilities to the focusing heuristics in her plan-recognition system ([Carberry, 1990b]) (described in Section 2.2 above). Bauer ([Bauer, 1994]) described a plan hierarchy closure method to attach Dempster-Shafer probabilities to any plan hierarchy.

Belief Networks

Several systems have used Bayes Nets to integrate uncertain inference into plan recognition. In Albrecht, Zukerman and Nicholson ([Albrecht *et al.*, 1998]), dynamic belief networks were trained to predict a user's goal based on observed actions in a multi-user dungeon video game. Horvitz and Paek ([Horvitz and Paek, 1999; Paek and Horvitz, 2000; Horvitz and Paek, 2000; Paek *et al.*, 2000]) use dynamic Bayesian Networks to recognize user intentions in several dialogue domains. Charniak and Goldman ([Goldman, 1990; Charniak and Goldman, 1991; Charniak and Goldman, 1993]) built an entire natural language understanding system, including plan recognition, in a unified dynamic belief network. Plan hypotheses were generated by piecing together evidence (previous utterances, plan roles of items in the current utterance, *etc.*) A priori probabilities of the likelihood of observations were then percolated through the network to determine the overall likelihood of the hypothesis.

Probabilistic Plan Libraries

Calistri-Yeh ([Calistri-Yeh, 1991]) suggests a simpler system of placing probabilities on edges in a Kautzian plan hierarchy. Combining this evidence allows an overall likelihood to be computed for a hypothesis within the plan space. Calistri-Yeh's system also supports recognition of a range of types of erroneous plans, with likelihoods computed for these erroneous plans as well.

Repairing Mistaken Inferences One topic that has typically been absent in the literature of plan-recognition systems for dialogue is this. What happens when the system makes (and commits to) the wrong hypothesis? Many of the natural language systems discussed above to not seem to have a repair mechanism. The system is based on the assumption that all previous hypotheses were correct. When a mistake has been made, probably the only thing many systems could do would be to re-recognize all of the previous input, keeping in mind the new evidence we have received.

Logical systems like Kautz' treat plan recognition as nonmonotonic reasoning, so incorrect inferences can be changed by new evidence. However, most implemented natural language systems, ([Carberry, 1990b; Lambert and Carberry, 1991; Ramshaw, 1991] for example), use heuristics and seem to have no mechanism for correcting incorrect inferences.

We believe that incorrect system inferences are an inevitable problem and show be addressed. Our plan-recognition system, which we describe in Section 4.3 below, attempts to address this, and the other shortcomings addressed in this section.

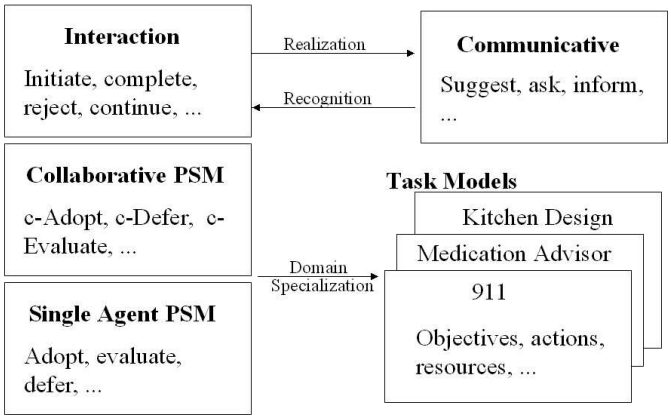

Figure 2: Collaborative Problem-Solving Model

3 Preliminary Model

This sections describes our preliminary collaborative problem-solving model ([Allen *et al.*, 2002]), which we have summarized graphically in Figure 2. On the right are several *task models* which are the only domain specific part of the model. These contain specific knowledge about the domain: what actions are considered objectives in the domain; what objects are considered resources; and knowledge about situations. They also include knowledge about actions and recipes in the domain. As opposed to most previous work, our model does not require that this task knowledge be static or mutually known. We only require that agents have the same understanding of what an action "means." This allows for richer collaboration between agents. Each can bring certain knowledge about the domain to the table, and by combining that knowledge they can build a recipe that neither could have produced separately.

A task model is used as a domain-specific "plug in" to the rest of the collaborative problem-solving model, which is domain independent. Objects from the task model are used as arguments for both the (single-agent) problem-solving (PS) and the collaborative problem-solving (CPS) levels. The problem-solving level involves single-agent problem solving, or changes that a single agent makes to its own *problem-solving state*. In like manner, the collaborative problem-solving level involves changes to the *collaborative problem-solving state* (jointly held by two or more agents).[7]

Although an agent can change its own problem-solving state, it cannot single-handedly change the collaborative problem-solving state. Doing so involves action from *all* of the

[7]At the current time, it does not seem to matter to our model whether elements in a collaborative problem state are decomposable into individual problem-solving-state elements ([Grosz and Kraus, 1996]) or if they are not ([Searle, 1990]).

agents involved. This means, in other words, that actions at the CPS level are *not* executable by a single agent. How is an agent to affect the CPS state then? It must do so through *interaction* with the other collaborating agents.

The interaction level contains individual actions which are used to negotiate changes in the CPS state. If all agents are amenable to the change and cooperate, a certain set of interaction actions will result in a change in the CPS state. Interaction acts are the next "meta" level up, as they take CPS-level actions as arguments.

Interaction-level actions (along with their CPS arguments) are what we use to represent intentions in utterances. However, as we have mentioned before, these intentions need to be communicated through the medium of language. An interaction act is realized by a *communicative act*. This realization process is what we earlier described as content planning. When a communicative act is "heard", it must be decoded to determine the interaction act behind it. We have now more precisely defined intention recognition under our model: the recognition of an interaction act from a communicative act.

In the rest of this section, we describe each individual "piece" of the model in more detail. We start first with task models. We then discuss the PS and CPS levels, the interaction level and finally the relationship between interaction acts and communicative acts. We then give several examples of this model applied to dialogues. After that, we discuss how this model overcomes the two main shortcomings of previous work: collaboration paradigms and interleaved planning and acting.

3.1 Task Models

A task model contains domain-specific knowledge that is used in the model. By saying our model is domain independent, we mean that the other levels besides the task model do not change from domain to domain. The model only requires what *type* of information needs to be included in the task model. Our theory is that collaborative problem solving is the same, regardless of the task and domain ([Allen *et al.*, 2000; Allen *et al.*, 2001b]). If agents are collaborating about travel, or about medication, or kitchen design, the collaboration is the same. A task model can be used in a "plug-and-play" fashion with the rest of the model. The task will change, but the collaborative problem-solving model remains the same.

A task model needs to contain the following types of information.

- *Objectives*: What actions are considered objectives in the domain. For example, in a rescue domain, objectives could include rescuing a person, evacuating a city, *etc.*
- *Recipes*: An agent's beliefs of how to do a (non-atomic) action. Although we do not adhere to any specialized definition of recipe, one example is that of Carberry's domain plan library (see above) which has action decomposition information about objectives. Also, this recipe library can be expanded or modified through agent planning (collaborative or not). We do not require, however, that collaborating agents have the same beliefs about recipes.

- *Atomic Actions*: What actions are in the domain and what exactly it means to perform the action. An agent need not know *how* to perform actions, but there must be agreement among agents of what actions *mean*.

- *Objects/Resources*: Knowledge about objects in the world as well as which objects are considered resources for the domain.

- *Situations*: The state of the world (or a possible world). An agent may only have partial knowledge about a given situation.

3.2 The Problem-Solving Level

The problem-solving level describes problem solving for a single agent. With this model, we are not trying to compete with formal models of agent behavior (*e.g.*, [Rao and Georgeff, 1991; Cohen and Levesque, 1990]). Rather, our single-agent model is a useful abstraction for modeling other agent's thought processes. "Under the hood," an agent can be designed in any way, as long as it is able to collaborate using this collaborative problem-solving model. It is necessary, however, for us to be able to reason about agent problem solving at this abstract level, since agents must eventually be able to collaborate about an individual agent's actions (see next section).

In general terms, an agent's problem-solving activities occur in three phases. Although certain constraints exist (*e.g.*, an agent has to have chosen an objective before it determines a recipe for the objective), the model does not impose linearity on these phases. An agent may revisit and change decisions made earlier. The three phases are the following.[8]

1. *Determining Objectives:* An agent may have many objectives to which it is committed which we call *adopted objectives*. However, at any one time, only a few of these objectives are actually be in focus and driving its behavior. We call these *active objectives*. An agent can at any time adopt new objectives, make adopted objectives active, abandon previously adopted objectives, *etc.*

2. *Determining and Instantiating Recipes for Objectives:* Once an objective has been determined, an agent must commit to a recipe it will use to attempt to obtain that objective. An agent may either choose a recipe from its recipe library (see *Task Models* above), or it may choose to *create* a new recipe via planning. Previously stored recipes can also be modified as the agent sees fit.

   Once a recipe has been committed to for an objective, an agent may choose to revisit that decision (perhaps after trying the recipe for a while but not making any progress). The agent can then abandon the current recipe choice and make another. Again, it is important to note that, as opposed to previous work (see above), this model allows

[8] Other research ([Wooldridge and Jennings, 1994; Wooldridge and Jennings, 1996; Wooldridge and Jennings, 1999]) has also included a phase of *team formation*, in which an agent finds a "team" with which to collaborate. Such a phase could account for such utterances as "Can I talk to you?" or "Could you help me with X?" Although this phase may eventually be added into our model, we leave this problem to future research. For this paper, we assume that agents already know they are collaborating.

the agent to revisit and/or change its decisions, rather than locking it into a decision once execution has begun.

3. *Acting:* As stated above, a recipe is something that tells the agent what to do next in a particular situation. This *next action* may be a subobjective, in which case the must choose whether to adopt it, choose a recipe for it, *etc.* Or, it may be an atomic action, in which case the agent must determine if it really wants to do it, and then execute it if it so chooses. There is no requirement that recipes be *correct* (*i.e.*, always ensure that the objective will be attained), so an agent needs to be able to reason about individual steps. It may be that the atomic action (or subobjective) returned by the recipe is determined by the agent's beliefs to be unproductive, in which case it may choose to modify or abandon the currently active recipe. In any case, in this phase, the agent reasons about what a recipe gives it, and acts accordingly.

It is important to note that our model does not impose any strict ordering on the phases above (besides the obvious constraints we mentioned previously). It does not specify *when* or *why* an agent might, for example, choose to reconsider its active objectives or to modify the current recipe. We believe that this freedom is an important part of allowing communication among heterogeneous agents. Requiring all agents to use the same *problem-solving strategy* seems too stringent a requirement. Different humans, certainly, use different problem-solving strategies. Even the same person may use different strategies in different situations. One person may be very cautious, constantly reevaluating his objectives, while another may be more efficient, having once made a decision, just working towards it without worrying if it was indeed the correct one.

Similarly, agents must be able to interact with agents which use completely different problem-solving strategies. We believe that this will not affect intention recognition, as language seems to be structured in a way which facilitates decoding the agents intentions, regardless of the problem-solving strategy.

Our problem-solving model consists of *objects* and *acts*. Objects serve as arguments to the acts. problem-solving acts can further be divided into acts relating to commitment and acts relating to reasoning. We discuss these areas separately below.

Problem-Solving Objects

The problem-solving objects were described earlier, and we list them again here to remind the reader. As this is work in progress, we do not claim this to be a complete set, although we do believe that the final set will be small.

- Objective
- Recipe
- Action
- Situation

- Resource

The problem-solving model and the task model fit together with problem-solving objects. In fact, problem-solving objects can be seen as being abstractions of task-model objects. In well-formed problem-solving actions, these will actually be instantiated by task-model objects (the *objective* may be loading a truck, *etc.*) These general problem-solving objects allow us to talk about problem solving in the abstract. More interesting are the acts which operate on them.

Acts Relating to Commitment

These acts change the agent's commitment to the various PS objects. The acts are listed here.[9]

- *Adopt*: Commits the agent to the PS object.
- *Select*: Moves an adopted object into the small set which is currently influencing the agent's behavior.
- *Defer*: Tables a selected object. The agent is still committed to the object, but it is no longer in the agent's immediate focus.
- *Abandon*: Removes an agent's commitment to an object before its purpose is fulfilled.
- *Release*: Removes an agent's commitment to an object which has fulfilled its purpose.

These are better understood in context with the PS objects to which they are applied. We describe the effect of these PS acts when applied to each of the PS objects.

Objectives The intentional "life cycle" of objectives is shown in Figure 3. An agent can **adopt** an objective, committing itself to attain it. As discussed earlier, an agent may have many adopted objectives, but only a small subset controls its current actions. By doing a **select**, an agent makes a certain objective active. An active objective can be demoted back to the adopted state by **defer**. At any time, an agent can **abandon** an objective, dropping its commitment about it. Finally, when an agent believes that an objective has been obtained, it may **release** it. Note that an objective is not automatically released when it is obtained. An agent must *believe* that it has been obtained and then consciously release it.

Recipes These do not quite fit the same model as objectives, since it does not make sense for an agent to be committed to more than one recipe for a single objective. An agent commits to a recipe for an objective by **adopting** it. An agent can also **release** or **abandon** an adopted recipe, similar to objectives. **Select** and **defer** do not play a role with recipes.

[9]We do not claim that this is a complete list. We do, however, believe that the final list will be short.

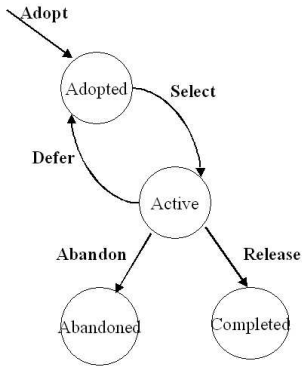

Figure 3: Intentional Life Cycle of Objectives

Atomic Actions These are similar to objectives. **Adopting** an action means that an agent is committed to performing it. A **selected** action is something that is being performed at the current time, which can be **deferred** (to suspend execution), **abandoned**, or **released**.

Resources These are objects that are somehow used in a recipe. They fit the same model as recipes in that only one resource is committed to for a certain slot in a recipe. Agents can **adopt** a resource, committing to use it in a recipe. They can also **abandon** or **release** a resource. **Select** and **defer** are not applicable to resources.

Situations These are somewhat similar to recipes. What is commitment to a situation? We believe this is used to explain "what if" type, possible worlds simulations. When an agent **adopts** a situation, it is committing to do reasoning according to that situation. **Abandoning** a situation reverts all reasoning to the *actual* situation. It is unclear what **releasing** a situation would mean. **Select** and **defer** also do not play a role with situations.

Summary Figure 4 gives a synopsis of applicability of commitment PS acts. An `x` shows that a PS act is applicable to the object and a blank indicates non-applicability.

Acts Relating to Reasoning

An agent must do reasoning to decide which commitments to adopt, abandon, and so forth. This is accomplished with the following reasoning PS acts.[10]

[10]Again, we do not claim this to be a comprehensive list, although we believe the final list will be small.

	Adopt	Select	Defer	Abandon	Release
Objective	X	X	X	X	X
Recipe	X			X	X
Action	X	X	X	X	X
Situation	X			X	
Resource	X			X	X

Figure 4: Application of Commitment PS Acts to PS Objects

Identify: Brings an object into focus. Used to determine which options are available.

Evaluate: Determines the goodness of an object in relations to its purpose.

Modify: Changes an object in some way.

We discuss these in the context of each PS object.

Objectives When an agent is considering choosing, releasing, abandoning, *etc.* an objective, it must **identify** one or more objectives that are possibilities. It can then **evaluate** an objective to determine if it's a good idea to adopt it (or select it, *etc.*) The agent can **modify** the objective in a certain way (make it more specific, change a parameter, *etc.*) before committing to it.

Recipes These are treated very similarly to objectives, although they are always considered *in relation* to an objective. An agent can **identify** or **evaluate** possible recipes for a specific objective. **Modifying** a recipe is essentially planning and allows for novel recipes to be formulated by the agent as necessity arises.

Atomic Actions The effect of reasoning PS acts on atomic actions are dependent on the context of whether the agent is planning (modifying a recipe) or acting. **Identifying** an action in planning identifies a possible action to be used in the recipe. In execution, it means that the agent queries the adopted recipe to identify the next step for execution. **Evaluating** an action in a planning context determines whether or not it is a good candidate for use in the recipe. In an execution context, an agent may choose to evaluate an action identified from the recipe before committing to actually perform the action. (There is no constraint that an adopted recipe actually lead to attaining the objective.) An action may be **modified** in either execution or planning.

Resources These can be **identified** as candidates for use in a recipe. An agent can then **evaluate** the goodness of a resource in relation to a recipe. It is not clear what it would mean to **modify** a resource.

	Identify	Evaluate	Modify
Objective	X	X	X
Recipe	X	X	X
Action	X	X	X
Situation	X	X	X
Resource	X	X	

Figure 5: Application of Reasoning PS Acts to PS Objects

Situations In all but the most trivial domains, an agent will not know the entire situation and will need to reason about it. **Identifying** a situation gives the agent more information about what the state of that world is. An agent **evaluates** a situation in order to decide if it is a desirable situation to be in. (This may also be helpful for an agent to decide whether or not an objective can be released.) Hypothetical situations can be **modified** as well.

Summary Figure 5 gives a summary of the applicability of reasoning PS acts to PS objects. Similar to Figure 4, `x` shows the applicability of acts to objects.

3.3 The Collaborative Level

Collaborative problem solving is an extension of single-agent problem solving. An agent team solves problems together much in the same way a single agent does, in the following stages.

- Determining Objectives
- Determining and Instantiating Recipes
- Acting

These phases are much more overt in multiple-agent problem solving, as agents must communicate and coordinate their reasoning. Because agents must do this collaboratively, these phases are much more visible. An important difference from single-agent problem solving is that no one agent can perform any collaborative activity alone.

Again at this level we have objects and acts, paralleling the single-agent PS model. In order to distinguish acts and objects at the two levels, we append a *c-* before any act or object at the collaborative level. These acts and objects have similar intuitive meaning to those at the problem-solving level and we will not redefine them here. There are some important differences, however, which seem to justify the conceptual separation of the two classes. We discuss these differences below. We first discuss collaborative problem-solving objects (CPS objects) and then collaborative problem-solving acts (CPS acts).

CPS Objects

The collaborative problem-solving objects are *c-objective*, *c-recipe*, *c-action*, *c-situation*, and *c-resource*. Again, these are abstractions of items at the task level, although there appear to be some differences (at least in some) from their problem-solving counterparts. We provide some preliminary thoughts here.

Grosz and Kraus ([Grosz and Kraus, 1996; Grosz and Kraus, 1999]) show that for an agent team to have a shared objective (goal), they must not only have individual intentions towards attaining that objective, but they must also have intentions not to interfere with the other group members' part of attaining that goal. The same would seem to hold with recipes, actions, and resources. Although more research is needed, c-situations may serve a function similar to common ground, or parts of the situation that all parties mutually believe.

CPS Acts

The intentional collaborative problem-solving acts are *c-adopt*, *c-select*, *c-defer*, *c-abandon*, and *c-release*. The reasoning CPS acts are *c-identify*, *c-evaluate*, and *c-modify*. CPS acts are definitely different from PS acts. They are actions which are composed by individual actions from the entire team. No single agent can perform a CPS act on its own. CPS acts have the effect of changing the collaborative problem-solving state.

Compatibility with SharedPlans

As we mention in Section 2.1, we believe that our model is compatible with the SharedPlans formalism. In fact, one way of looking at our model is an elaboration of some of the key operators (such as Elaborate_Group, or Lochbaum's *communicate* recipe) in the SharedPlans framework. Adoption, evaluation and so forth of objectives would reside at a higher level than SharedPlans, which assumes that a high-level goal has already been chosen. The adoption, evaluation, *etc.* of recipes is a further specification of the Select_Recipe_GR operator. Most other acts, such as adopting resources, evaluating actions, and so forth, provide more details of the Elaborate_Group level.

Grosz and Kraus themselves ([Grosz and Kraus, 1996]) acknowledge the need for further refinement of the SharedPlans formalism, especially at the Elaborate_Group level. We believe that our research provides this further refinement.

3.4 The Interaction Level

CPS acts are generated by the individual *interaction acts* of agents in the team. This part of our model is perhaps the least well formulated. We describe our preliminary model here. Again, completing this part of the model is part of the research plan described in Section 5.

An interaction act is a single-agent action which takes a CPS act as an argument. The interaction acts are *initiate*, *continue*, *complete* and *reject*. These are defined by their effects and are similar to the grounding model proposed in ([Traum, 1994]).

When a CPS act is proposed, a certain amount of refining the details of the act happens and then the CPS act is accepted by each member of the team, at which time the act is generated. The very first interaction act (the one that starts this process off) in an *initiate*, the final act (which generates the CPS act) is a *complete*, and all acts in the middle are *continues*. A *reject* by any member of the team causes the CPS act to fail.

Because this part of the model is only at a very preliminary stage, we provide some sample dialogues which serve to demonstrate various phenomena we have encountered thus far. We then discuss how social paradigms fit into this model.

Initiate-Complete The simplest interaction is an initiate-complete pair. Consider the following dialogue.

U1: "Let's go to the park."
S1: "OK."

Here U1 is an `(initiate (c-adopt (c-objective 'U1 & S1 go to park')))` and S1 is a `(complete (c-adopt (c-objective 'U1 & S1 go to park')))`. After S1, the `(c-adopt ...)` is generated, and U and S now have a collaboratively adopted c-objective to go to the part together.

Initiate-Reject The following is an example of an initiate followed by a reject.

U1: "Let's go to the park."
S1: "No."

In this case, S1 is a `(reject ...)` and the `(c-adopt ...)` fails. The collaborative problem-solving state is not changed.

Initiate-Continue-Complete An initiate may not contain enough detail to be accepted. Consider the next example.

U1: "What should we do?"
S1: "Let's go to the park."
U2: "OK."

Here U1 is `(initiate (c-adopt (c-objective ??)))`, where the details of the c-objective are not present. S continues the interaction (S1) by providing the details, which U then completes (U2).

Initiate-Continue*-Complete Of course, more complex interactions can occur. The following interaction has several continues before it is completed.

User	**Let's rescue the heart attack victim at Marketplace Mall.**	(1)
	`(initiate (c-adopt (c-objective (rescue person1))))`	
System	**OK**.	(2)
	`(complete (c-adopt (c-objective (rescue person1))))`	
User	**Let's send ambulance 1 to take them to Strong hospital.**	(3)
	`(initiate (c-adopt (c-recipe (take-to-hospital))))`	
	`(initiate (c-adopt (c-resource (vehicle amb1))))`	
	`(initiate (c-adopt (c-resource (hospital Strong))))`	
System	**OK.**	(4)
	`(complete (c-adopt (c-recipe (take-to-hospital))))`	
	`(complete (c-adopt (c-resource (vehicle amb1))))`	
	`(complete (c-adopt (c-resource (hospital Strong))))`	

Figure 6: A Planning Dialogue

U1: "What should we do?"
S1: "Let's go to somewhere."
U2: "Let's go to a park."
S2: "Let's go to a big park."
U3: "Let's go to Ellison Park."
S3: "OK."

Each continue provides more details on the objective, until finally it is agreed upon.

3.5 Interaction Acts and Communicative Acts

The final part of the collaborative problem-solving model is *communication acts*. As we have discussed above, interaction acts are represented at an intentional level. In order for communication to occur, these intentions need to be transformed into language (communicative acts) and then back again at the other end (agent). These are the respective processes of content planning and intention recognition.

3.6 Examples

In this section, we give several examples in order to demonstrate the dialogue model. The representation of interaction acts in the dialogues is not the formal representation we use in our system. The representation given here is sufficient, however, to show which interaction and CPS acts are intended by each utterance. The actual TRIPS system uses an event semantics to represent knowledge.

A Planning Example

Figure 6 gives an example planning-only dialogue from a 911 operator domain where the user and system are working together to deal with 911 emergencies. For each utterance,

System	**You should take your Zoloft now.**	(5)
	`(initiate (c-select (c-action (take user Zoloft))))`	
User	**Where is the Zoloft?**	(6)
	`(continue (c-select (c-action (take user Zoloft))))`	
	`(initiate (c-identify (c-situation (loc Zoloft ?x))))`	
System	**On the counter.**	(7)
	`(continue (c-select (c-action (take user Zoloft))))`	
	`(continue (c-identify (c-situation (loc Zoloft counter))))`	
User	**OK. [takes pill]**	(8)
	`(complete (c-identify (c-situation (loc Zoloft counter))))`	
	`(complete (c-select (c-action (take user Zoloft))))`	

Figure 7: An Acting Dialogue

the corresponding interaction acts are listed below it.[11]

At this point of the dialogue, there are several pending emergencies to be dealt with, including a heart-attack victim at Marketplace Mall. The user decides to initiate an adopt-objective and does so with utterance 1. The system completes this act with utterance 2. (This dialogue is also a good example of a master-slave paradigm, since the system simply accepts all of the user's proposals.)

Utterance 3 initiates several acts (each of which needs to be addressed in the response in utterance 4) initiating adopting a recipe and two resources. Again, the system completes these with utterance 4.

An Acting Example

Figure 7 gives an example of a dialogue about only acting from a Medication Advisor domain. In this domain, the system helps a person manage his medication regimen. In this example, the user and system already know that the user has a prescription for Zoloft.

In utterance 5, the system initiates a select-action for the user to take a Zoloft. Recall from Section 3.2 that selecting an action means to actually start doing it. (Note that the action is already adopted; since the user has a prescription, he is presumably already committed to performing the action.) The user, instead of immediately accepting, initiates an identify-situation in order to find the location of the Zoloft. Because this neither accepts nor rejects the select-action, it continues it.

In utterance 7, the system identifies the location of the Zoloft for the user. This is a continue and not a complete because the system has added new information to the proposed CPS act. It is not completed until utterance 8 where the user accepts the answer. The user also completes the select-action in utterance 8 and then begins performing the action.

[11] Because of space requirements, we have omitted some interaction acts here, such as *c-identify*, which are accepted in each case and do not play a large role in these particular dialogues.

User	**Send ambulance one to Parma right away.**	(9)
	`(initiate (c-adopt (c-action (send amb1 Parma))))`	
	`(initiate (c-select (c-action (send amb1 Parma))))`	
System	**OK.** [**sends ambulance**]	(10)
	`(complete (c-adopt (c-action (send amb1 Parma))))`	
	`(complete (c-select (c-action (send amb1 Parma))))`	
System	**Where should we take the victim once we pick them up?**	(11)
	`(initiate (c-adopt (c-resource (hospital ?x))))`	
User	**Rochester General Hospital.**	(12)
	`(continue (c-adopt (c-resource (hospital Strong))))`	
System	**OK.**	(13)
	`(complete (c-adopt (c-resource (hospital Strong))))`	

Figure 8: Interleaved Planning and Acting Dialogue

System	**Let's schedule a time for you to take your Prinivil today.**	(14)
	`(initiate (c-adopt (c-objective (schedule Prinivil))))`	
User	**No.**	(15)
	`(reject (c-adopt (c-objective (schedule Prinivil))))`	

Figure 9: Rejection Dialogue

An Interleaved Planning and Acting Example

Figure 8 shows an interleaved planning and acting dialogue. At this point in the dialogue, the user and system have already adopted the objective of rescuing a victim in Parma by the recipe of taking him to a hospital.

Utterance 9 initiates both the adopt and the select (*i.e.* adopt and begin immediately) of the action of sending the ambulance to Parma. The system completes these acts in utterance 10. Note that this acting was begun *before* the user and system had a complete plan for rescuing the victim.

Utterances 10–13 deal with adopting the resource of a hospital to which take the victim.

A Rejection Example

Figure 9 is a short (and blunt!) example of an agent's ability to reject a proposal. The system makes a proposal in utterance 14, which the user refuses in utterance 15. At this point, the proposed CPS act fails and the collaborative problem-solving state does not change. Rejection is an important feature of non-master-slave collaboration paradigms.

A Negotiation Example

Figure 10 shows an example of a negotiation within our dialogue model. Negotiation is not needed in master-slave dialogues, since the slave cannot negotiate. However, in most

User	**Let's use Ambulance one.**	(16)
	`(initiate (c-adopt (c-resource (amb1))))`	
System	**Ambulance one is too far away.**	(17)
	`(reject (c-adopt (c-resource (amb1))))`	
	`(initiate (c-evaluate (c-resource (amb1) bad too-far)))`	
System	**Ambulance two is closer.**	(18)
	`(initiate (c-evaluate (c-resource (amb2) good closer)))`	
System	**Let's use it.**	(19)
	`(initiate (c-adopt (c-resource (amb2))))`	
User	**OK, we'll do that instead.**	(20)
	`(complete (c-evaluate (c-resource (amb2) good closer)))`	
	`(complete (c-evaluate (c-resource (amb1) bad too-far)))`	
	`(complete (c-adopt (c-resource (amb2))))`	

Figure 10: Negotiation Dialogue

other collaboration paradigms, negotiation can be an important part of dialogue.

In utterance 16, the user proposes adopting ambulance1 as a resource. The system rejects the proposal and initiates its evaluation that ambulance1 is not a good resource because it is too far away. It then gives the evaluation that ambulance2 is a good resource since it is closer. In utterance 19, the system then initiates an adopt-resource of ambulance2. In utterance 20, the user accepts this proposal along with the two evaluations by the system.

3.7 Modeling Collaboration Paradigms

Most previous research restricted dialogue coverage to the master-slave collaboration paradigm, where one agent (either the user or the system) completely controls the dialogue and the other agent is compelled to accept all its proposals.

Because we stipulate that CPS acts cannot be directly executed by a single agent, agents have the ability to negotiate the contents of an act or to simply refuse it. At one extreme, this can model the mixed-initiative paradigm, where both agents are equal parters in the collaboration. At the other extreme, it can also model the master-slave paradigm, since it does not *require* agents to negotiate or refuse acts. In addition, the model can cover the entire spectrum between these extremes. This is an important and key feature of our model which allows it to account for a much wider variety of discourse phenomena than previous work.

3.8 Modeling Collaboration about Planning and Acting

As we discussed in Section 2, previous models covered collaboration about planning or collaboration about acting, but not both. This restricts the types of dialogues which can be covered to domains where *only* planning or *only* acting occurs.

We explicitly model collaboration about both planning *and* acting and allow for agents to interleave them. This allows us to model collaboration about planning, acting, or planning and acting.

4 Preliminary Results

In the last section we described our preliminary collaborative problem-solving model. In this section, we report preliminary results which we have attained using this model. The preliminary work described here is the base from which future research (see Section 5) will take place.

We are using the TRIPS system ([Ferguson and Allen, 1998; Allen *et al.*, 2000; Allen *et al.*, 2001a; Allen *et al.*, 2001b]) as a testbed for our model. We first give an overview of the TRIPS system. We then discuss use of the problem-solving model in the TRIPS Medication Advisor. Finally, we discuss a plan-recognition algorithm we have implemented, which will serve as a base for the intention-recognition implementation for the model.

4.1 TRIPS Overview

TRIPS, The Rochester Interactive Planning System, is a conversational agent designed to collaborate (in a mixed-initiative paradigm) with a human user in a variety of domains. One of the goals of the TRIPS project is to build a domain-independent dialogue-system *core*, which can be easily ported to new domains. Thus far, TRIPS has been successfully ported to several very distinct domains. We discuss some of those domains below. We then give an overview of the TRIPS architecture and show how intention recognition is utilized within TRIPS.

TRIPS Domains

One of the goals of the TRIPS project is to build a domain-independent dialogue-system core which can be easily ported to new domains. Thus far TRIPS has been ported to several very distinct domains. We mention several of the domains here.

TRIPS Pacifica Emergency evacuation. Pacifica (a fictitious island) must be evacuated because of an approaching hurricane.

TRIPS Monroe Emergency 911 operation. This domain is set Monroe County, New York. The user and system are operating a 911 emergency facility and must make and execute plans to respond to various emergencies which come up.

TRIPS Medication Advisor The system assists the user to understand and follow a medication regimen. It also answers questions about regimen adjustment, problems which come up, side-effects, *etc.*

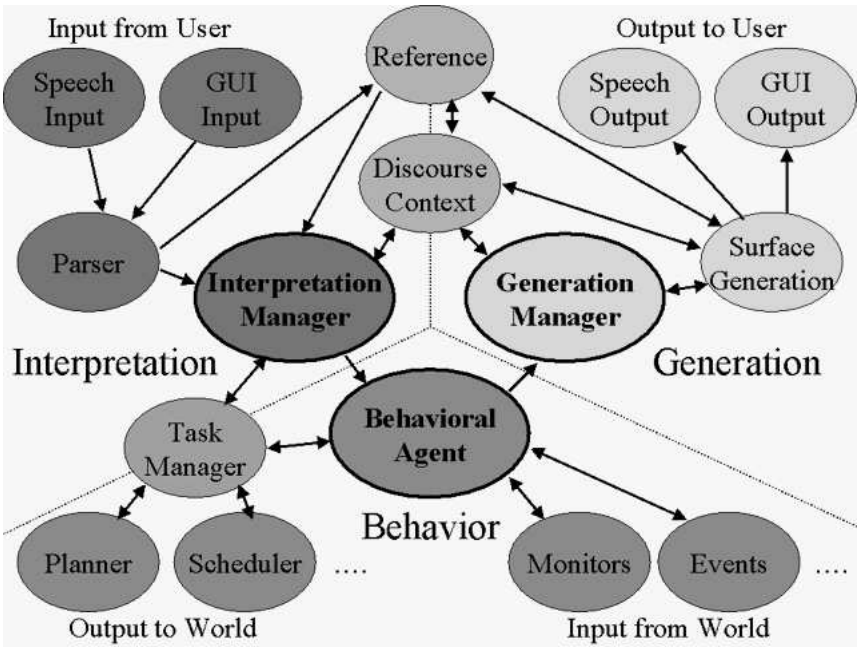

Figure 11: TRIPS Architecture

The above systems are quite different domains, and the porting to them has helped ensure that the core truly is domain independent. We believe this is a good opportunity for our collaborative problem-solving model. Applying it to distinct domains will ensure that it, too, is truly domain independent.

It is also important to note that the TRIPS Monroe domain is a perfect example of a domain which requires interleaved planning and acting (not to mention replanning when things go wrong). Because of the time sensitive nature of a 911 domain, it is sometimes best to start execution of a plan (such as sending an ambulance to a victim) and later finish specifying other details of the plan (such as which hospital to take the victim to).

TRIPS Architecture

TRIPS ([Allen *et al.*, 2001a; Allen *et al.*, 2000])[12] has a modularized, asynchronous, agent-based, message passing architecture. Figure 11 shows a generalized view of the architecture.

As shown in Figure 11, multi-modal input comes in to the system either through speech recognition, the keyboard, or other GUI interaction. This input is then passed to the Parser, which outputs a list of (direct[13]) speech acts which describe the input.

[12]This section only provides very high-level details of the system (those needed to understand the interaction of intention recognition). For more details about the architecture see our webpage at *http://www.cs.rochester.edu/research/cisd/*.

[13]As opposed to indirect speech acts ([Searle, 1975]), the processing of which is described below.

These speech acts are then sent to the Interpretation Manager, which is the main interpretation agent of the system. The Interpretation Manager, attempts to resolve all references in the input (via the Reference Manager), and then, using the method of Hinkelman and Allen ([Hinkelman and Allen, 1989; Ferguson *et al.*, 1996]) computes a set of preliminary indirect speech acts that the input could account for. Each of these possibilities is sent to the Task Manager (more details below) for more processing and evaluation. It is in the Task Manager that intention recognition takes place. For each hypothesis, the Task Manager returns to the Interpretation Manager an full interaction act, with accompanying collaborative problem-solving act and a score of the goodness of the hypothesis.

The Interpretation Manager then passes the best interpretation on to the Behavioral Agent. This autonomous agent then deals with the input as it sees fit (*i.e.*, by accepting and performing the request, tabling the request in order to perform some more urgent action, *etc.*) The Behavioral Agent then communicates its intentions and/or results to the Generation Manager, whose job it is to communicate this back to the user in a socially and linguistically coherent manner.

Again, this has been just a quick overview of the system architecture. But, it gives us enough detail to be able to discuss in more detail how intention recognition fits into the system by describing the Task Manager.

The Task Manager The Task Manager, as one of its many jobs, performs intention recognition in the system. The purpose of this section is not to describe the actual intention recognition (which is only partially implemented), but to discuss the *interaction* of the intention recognition with the rest of the system.

As stated above, the input to the intention recognizer is an (indirect) speech act hypothesis (generated by the Interpretation Manager), and the output is the full interaction act the user intended with the speech act, and a score of goodness, which the Interpretation Manager uses to decide which interpretation of the input is the best. It then sends this off to the Behavioral Agent.

As stated above, the intention recognition of the Task Manager is only partially implemented. In the TRIPS Medication Advisor, we have analyzed a dialogue according to our model and built the system actually using those acts as message traffic, the details of which we describe in the next section. We have also implemented a plan-recognition system (described below) which will serve as a module of the intention-recognition system.

4.2 Collaborative Model in TRIPS Medication Advisor

Recently, we used our preliminary collaborative problem-solving model in the implementation of the TRIPS Medication Advisor. Although the intention recognition is not yet implemented, the use of the model in the system is significant in two ways. First, it demonstrates that this model could be applied to a complex dialogue (we give examples below). Secondly, it showed that the rest of the system (Behavioral Agent, Generation Manager, *etc.*) could work (*i.e.*, make decisions, plan content, *etc.*) under this model. We now have everything in place to implement an intention-recognition system for the model (see research plan below).

U1	Good Morning
S1	Good morning
U2	Can I take an aspirin
S2	No, you are taking Celebrex and Celebrex interacts with aspirin
U3	Oh
U4	Could Celebrex cause headaches
S3	No, headaches are not an expected side effect of Celebrex
S4	Do you have a headache?
U5	Yes
S5	How bad is your headache on a scale of 0 to 5?
U6	One
S6	Your doctor says you can take Tylenol for mild headaches
U7	OK. I'll try that
U8	Why am I taking Celebrex
S7	You are taking Celebrex to help your arthritis
U9	Should I take one now
S8	Yes, you take one every morning
U10	Is there anything else I need to take now
S9	Yes. You need to take your Prinivil
U11	Which one is that
S10	The yellow one [displays picture]
U12	OK
U13	Could that be causing my headache
S11	Yes, it could.
S12	But the tylenol should help.
U14	OK. thanks

Figure 12: TRIPS Medication Advisor Dialogue

The Medication Advisor dialogue illustrates several interesting dialogue phenomena which our model is able to handle. The dialogue is shown in Figure 12. We discuss here only some of the more interesting points of the dialogue, which illustrate the coverage of our model.

Utterance U2, at the start of the dialogue is analyzed as a *initiate* of a *c-evaluate* of a *c-action*, namely that of the user taking an aspirin. The user is considering doing the action, and wants first to evaluate it. The system *completes* the *c-evaluate* by giving the action a bad evaluation. So far, the user has had the problem-solving initiative.

At utterance S4, the system has recognized that the user may have an objective of curing a headache, so the system takes initiate in the dialogue by doing a *initiate* of a *c-identify* of a *c-situation* to determine if the user has a headache. After determining this, the system, in utterance S6, *initiates* a *c-adopt* of a *c-recipe*, that of taking Tylenol to cure the headache. The user accepts and *completes* with U7.

Later in the dialogue, at U13, it appears that the user is looking for what medicine may be causing the headache, in order to *c-adopt* a new *c-recipe* for getting rid of it, namely stopping taking the medicine causing the headache. The system notices this and in S12 *initiates* a *c-adopt* of the *c-recipe* discussed before, that of taking Tylenol.

As can be seen by this one dialogue, even a seemingly information-seeking only dialogue can be analyzed at a deeper problem-solving level. Our problem-solving model is able to effectively handle such phenomena as system initiative, action evaluation, and the like. This, in turn, allows us to build more proactive, helpful systems.

4.3 Implementation of Plan-Recognition Algorithm

In Section 2.3, we described some of the major shortcomings of previous work in plan recognition. Intention recognition is a subset of plan recognition, and most intention-recognition systems utilize plan-recognition algorithms. Part of our research plan is to implement an intention-recognition algorithm for our model (see Section 5 below), and we must be able to adequately deal with these issues.

This section describes our implementation of a plan-recognition system which partially overcomes the shortcomings of speed, ambiguity and mistaken system inference [Blaylock, 2001]. We will first describe the plan-recognition system and then how it addresses each of these issues. In the Section 5.2 we discuss plans to extend this system to handle intention recognition for our collaborative problem-solving model.

Plan Recognizer Overview

In this section we give a basic overview of the plan recognizer. Many of the implementational details are beyond the scope of this paper, but they can be found in [Blaylock, 2001].

Plan Representation Plans are represented hierarchically, similar to that of Carberry ([Carberry, 1990b]). In addition to recipe information in the plan library we also explicitly

```
(RescuePersonFrame
     :goal (at-loc ?p ?h)
     :resources ((Vehicle ?v) (Hospital ?h))
     :input-vars ((Person ?p) (Ailment ?a) (Location ?l))
     :constraints ((Hospital ?h) (Person ?p) (Vehicle ?v)
                   (Location ?l) (Ailment ?a) (has ?p ?a)
                   (treats ?h ?a) (at-loc ?p ?l))
     :focused-solution S1
     :solutions ((:id S1
         :agreed-actions
           NIL
         :expected-actions
           (((move ?v ?l) NIL)
            ((load ?p ?v) ((at-loc ?p ?l) (at-loc ?v ?l)))
            ((move ?v ?h) ((in ?v ?p)))
            ((unload ?p ?v) ((in ?v ?p) (at-loc ?v ?h)))))))
```

Figure 13: The `RescuePerson` Objective Frame

represent the actual details of the plan which is being built. As Allen notes ([Allen, 1997]), it is necessary to save information about the plan separately from just a recipe library in mixed-initiative planning contexts where the steps of the plan are not being immediately executed. This stored information allows agents to revisit previously discussed portions of the plan, and to eventually execute it.

Plans are stored as a tree of *objective frames*. Each objective frame contains a recipe for a certain objective, and is augmented with other information that aides in plan recognition.

Figure 13 shows an example of an uninitialized objective frame from the TRIPS-Monroe domain. This frame represents information about a recipe for rescuing a person. In this frame, the `:goal` section describes the objective of the frame. The `:resources` section describes what objects are considered resources for this objective. The `:input-vars` section describes the objects used in this objective which are not resources. The `:constraints` section describes *frame constraints*, general constraints that must be true before execution of the plan begins. It gives variable type constraints as well as general plan constraints (such that the hospital in the solution must be one that treats the persons ailment).

The `:focused-solution` section describes which solution in the `:solutions` section is committed to at this time. This allows the storage other solutions which may have been discussed previously, in order to answer such queries as "What was the other solution we had talked about?"

Each solutions in the `:solutions` section consists of `:expected-actions` and `:agreed-actions`, as well as *action constraints*, which are constraints that must be true before this particular action is executed. The `:agreed-actions` section contains actions which have been discussed and committed to. The `:expected-actions` section contains steps in the recipe

which are expected but not yet committed to.

Recognition Algorithm The system handles both descriptions of goals (such as "Let's get Bob to Strong hospital") as well as descriptions of actions ("Move ambulance one to Bob's house"). These roughly correspond to top-down and bottom-up dialogue strategies, respectively ([Carberry, 1990b]). Goal descriptions are matched by searching through possible objective frames on the `:goal` field. It is fairly straightforward and uninteresting and will not be discussed here further.

When an action has been mentioned, the system tries to find a matching action in `:expected-actions` in an objective frame solution. If the actions can be unified, instantiating the variables within the objective frame, without any constraint violations, then a match has been found. The action, and all actions above it[14] are moved from the `:expected-actions` list to the `:agreed-actions` list.

Of course, there may be *many* matches. Any action may match into many different objective frames, as well as matching several times within the *same* objective frame (in our `RescuePerson` objective frame, a `move` action matches two of the four steps). There may also be many initialized (*i.e.*, partially instantiated) objective frames in which matches are also found. The system uses context to constrain this ambiguity.

Plan recognition is typically categorized into two kinds ([Cohen *et al.*, 1982]), which characterize the attitude of the agent being observed. *Keyhole recognition* occurs when an agent is unaware that it is being observed (or indifferent about the observation). There is no attempt on the observed agent's part to help (or hinder) the plan recognition. *Intended recognition* occurs when the observed agent is purposely trying to structure actions in a way which clearly transmits its plans to the observing agent.[15] Natural language discourse is a case of intended recognition. In order for communication to proceed successfully, the "receiving" agent must be able to understand what was meant by the utterance.[16]

The fact that communication is a form of intended recognition allows us to use context to constrain our search. We discuss here how the algorithm searches for action matches.

As a dialogue proceeds, many objective frames are potentially discussed. When all of the actions of its solution have been moved to the `:agreed-actions` section, an objective frame is said to be *completed*. Objective frames which have been discussed (and partially instantiated), but are not completed are said to be *pending*. The frame into which the most recent action was matched is called the *current frame*.

Within each frame, actions in the `:expected-actions` section are searched in order. Search among frames proceeds in the following order. First the system checks for matches

[14]The actions in a solution are implicitly temporally ordered.

[15]A third type of plan recognition occurs when the acting agent is trying to thwart recognition of its plans. Pollack ([Pollack, 1986a]) calls this an *actively non-cooperating actor*. Very little research has been done for this third type of recognition, which may be why it is frequently not included in the typology.

[16]Note that even in situations where one agent is trying to deceive the other, it is still intended recognition. The *content* of the message may be deceitful, but the "sending" agent *intends* for the "receiving" agent to understand the content (although not the fact that the content is false).

within the current frame. It then searches through all pending frames, in most-recently-mentioned order. When there is still no match, it matches all possibilities for *new frames* within the objective frame library.

The search stops when the first action is found that unifies without violating any objective frame constraints. The only exception to this is if we are searching for a new frame, in which case we return all frame types which have a match (action matching within solutions however, is still stopped at the first match, even if there are multiple matching actions in the solution).

Of course, in stopping on the first found solution, there is potentially a lot of search space that is not explored. The intended nature of communication, however, means that the "sending" agent *wants* the "receiving" agent to understand the message. This leads us to the following two hypotheses:

1. The "sending" agent will attempt to make the message easy to decode (compute).
2. The "sending" agent will attempt to make the message unambiguous.

Without context, language is inherently very ambiguous. It is our hypothesis that the mutually understood language encoding (discourse structure) includes a mutually known search strategy to help ease computation and alleviate ambiguity. Thus, the first match encountered is likely to be the match the system was *intended* to find.

Of course there is always the possibility that a match may be wrong. In order to handle this possibility, our system must be able to do two things: (1) detect the error and (2) repair it. The system recognizes that it has previously made a match error when the user mentions a new action, the effect of which violates an action constraint for any previously mentioned action (in any object frame). On each new input, the system searches `:agreed-actions` in *all* previously mentioned frames (both completed and pending) for action constraint violations. When a violation occurs, this means that under our current interpretation some earlier solution will be rendered inoperable by the effects of this current action. Assuming that the user is not making a mistake,[17] this means that the system has made an error.

In order to correct the error, the system uses an undo mechanism, which undoes one action at a time, until it reaches a point where the action constraint is no longer violated by the current action. Once the backtracking is complete, the search process is restarted, beginning with the last undone action, this time continuing the search from the previous, incorrect hypothesis. The system then reinterprets all undone actions (being careful not to violate the problem action constraint). Thus, as needed (as errors are detected), the algorithm backtracks and explores the search space further. Eventually, if there is a solution, this algorithm will find it (provided it detects the error).

This algorithm is actually very similar to what Schmidt *et al.* ([Schmidt *et al.*, 1978], also Section 2.3 above) report that humans did in experiments. Humans would come to just one conclusion and then revise it if later information invalidated it.

[17]Both Pollack ([Pollack, 1986a; Pollack, 1986b; Pollack, 1987; Pollack, 1990]) and Quilici ([Quilici, 1989]) have shown that the assumption that the user is always correct is a limiting one. It may be possible to use a heuristic (based on how far the system has to backtrack, *etc.*) to determine if the system should query to make sure the user isn't the one who made the mistake.

Addressing Previous Issues

As we discussed in Section 2.3, previous work in plan recognition has had several deficiencies, including slow runtime, lots of ambiguity, and the inability to recover from mistaken inferences. We describe here how our plan recognizer deals with these issues.

Speed Speed problems with previous plan recognizers were mostly caused by having to search such a large space for possible explanations. Although in the worst case our system will also search the entire space, it only does so if it must (most other systems *automatically* search the entire space and generate *all* possible hypotheses). Because communication uses intended recognition, we believe that the number of hypothesis revisions will be minimal. Unfortunately, this only solves part of the problem. We still do not have any mechanisms for dealing with ambiguity when an action matches multiple *new* frames.

Ambiguity Because the recognizer stops at the first match, we do not have to reason with ambiguous results when working with previously mentioned objective frames. As mentioned above however, the problem of ambiguity still exists when instantiating new frames.

Mistaken Inferences Our system explicitly deals with mistaken inferences through use of a hypothesis revision mechanism. If the system has previously committed to an incorrect result, it can reinterpret the data and form a new hypothesis.

5 Research Plan

In previous sections we described our preliminary collaborative problem-solving model, as well as results we have obtained up to this point. However, as can be seen, there is still much work to be done. In this section, we describe our proposed plan of research, which consists of three general areas.

1. Complete the collaborative problem-solving model
2. Develop an intention-recognition system which utilizes the mode
3. Evaluate the model and intention-recognition system

We discuss each below, commenting on future work to be done. We then give a rough research schedule.

5.1 Completing the Model

The model we discussed above is preliminary. There are still issues to be resolved in order to solidify it. We mention many of these issues here.

Act Applicability Figures 4 and 5 show the applicability of problem-solving acts to problem-solving objects. As we mentioned above, there is still some question as to the status of some of these, including selection of recipes, release of situations, *etc.* Further defining the place of recipes, situations, and resources should make this more clear.

Interaction Level Improvement The interaction level is still fairly undeveloped. We desire to revisit the level in light of other research ([Sidner, 1994a; Sidner, 1994b]) to see what, if any, modifications are necessary.

Status of Grounding Part of the interaction level will most certainly involve some sort of problem-solving grounding. Participants need to mutually know that the interaction has taken place in order for the collaborative problem-solving actions to be generated.

One Utterance, Multiple Acts In our preliminary dialogue annotation for the TRIPS Medical Advisor, we discovered that a single utterance can implicitly carry multiple intentions. For example, an *initiate* of a *c-evaluate*, in the right context, can also be a *c-initiate* of a *c-adopt*. The exact meaning of these multiple acts, including if they can be individually accepted, rejected, *etc.* still needs to be researched.

More Acts? We also need to determine if the model is incomplete, *i.e.*, if it is missing problem-solving acts or objects. We will annotate dialogues from several different domains in order to try to make the model as complete and independent as possible.

5.2 Developing Recognition Algorithms for TRIPS

An important part of our research will be to implement a domain-independent intention-recognition system based on this model to be used in the TRIPS system. We discuss here several issues for such an implementation.

Ensuring Domain Independence It is difficult to claim, or actually have, domain independence when something has only been tried in one domain. The TRIPS system is a perfect place for an implementation because it is meant to be a domain-independent dialogue-system core (see discussion above). In order to ensure domain independence, we will use and test the intention recognition in both the Medication Advisor and Monroe-911 domains of TRIPS. In our experience so far, these seem to be very dissimilar tasks at an intention-recognition level. Testing in both domains should reveal domain-dependent assumptions the implementation has made and allow us to fix them.

Task-level Plan Recognition Section 4.3 describes our currently implemented task-level plan-recognition system. This system is domain independent and only requires a task-specific plan library for its knowledge of the domain. Currently, however, it only recognizes user requests for actions or declarations of goals. We intend to expand this system to be

used for task-level intention recognition, which would then be processed at a more abstract collaborative problem-solving level. We will need to include reasoning capabilities about task-level resources and situations. There may also be some task-level evaluations or the like which would also have to be included.

Expected Discourse Moves As we discussed above, task-level plan recognition already has expectation built into it. We will use discourse cues and collaborative structure to utilize expectation at both the interaction and CPS levels. For example, we would not expect an INITIATE to be ignored by the other party. Instead we would expect *something* in response to the INITIATE, whether it be a COMPLETE, a REJECT, or a CONTINUE of the proposed CPS act. At the CPS level we may be able to build some typical CPS recipes, which we could use much in the same way domain-level recipes are used with agreed actions and expected actions.

We will also make use of various discourse cues to notice when expectations are likely to be violated. An important use of discourse cues in dialogue is to signal unexpected discourse moves. Of course, we must find a way to know, probably by the content of the discourse cues, where the algorithm should look for a match, if the expectation is to be violated. Using discourse cues under this new model may give us new significant insights into their meanings.

Domain-Independent Directives In addition to cue phrases, there seem to be some language which speaks directly to higher levels and not to the task level. Consider the utterance "Let's use the helicopter." The word "use" here is indicating a resource, regardless of the domain. We will attempt to find and make use of such domain-independent language.

Ambiguity and Speed Although our plan-recognition system uses a first-is-best search strategy, there are still ambiguity and speed issues which need to be dealt with. An especially important area is when the algorithm does not find a match anywhere in the pending objective frames. In this case, focus is shifting to some new, still uninstantiated objective frame which must be found.

The current algorithm simply searches the entire frame library for *all* possible matching frames. This is somewhat slow (linear in the number of frames), but more importantly, it potentially causes ambiguity. We will need a strategy for handling ambiguity in a way that takes the following into account:

1. *Immediacy:* Because we are in a dialogue setting, the TRIPS system usually uses our output immediately in formulating a response. Sometimes a wait-and-see approach will work, but other times the system will need to be able to make a decision.

2. *Speed:* It is impractical to follow several ambiguous "trails" for very long. After several turns, the ambiguity can potentially balloon, slowing down the system considerably.

A somewhat straightforward, although not perfect solution is the use of clarification (see Section 2.3). Too many clarification questions, however, can lead to a breakdown in

communication. A more intelligent use of clarification would be to employ a wait-and-see strategy, only utilizing clarification if the ambiguity level balloons or if the TRIPS system really needs an answer. Deciding, however, when the system needs an answer may be a difficult problem.

5.3 Evaluation

Evaluation is an important part of any research. In order to evaluate the collaborative problem-solving model, we propose to train annotators who will annotate dialogues in a variety of domains with the model. This will demonstrate not only the domain independence of the model, but also test how closely annotators can agree with each other in annotation.

Another important verification of the model will be the building of an automated intention-recognition system, and using it in several distinct domains. We believe we will be able to give examples of many discourse phenomena handled by our model, which previous systems were unable to handle.

5.4 Research Timetable

Below we give a rough schedule for future research. Because it is difficult to determine how long any one research activity will take, we describe the plan only in "phases." We believe that each phase will correspond to about a semester.

Phase 1 Unofficial annotation of dialogues to help complete model and give insights for later annotation experiment. Begin work on intention-recognition implementation. Complete model specification.

Phase 2 Write annotation manual based on finalized model. Conduct annotation experiments. Continue intention-recognition implementation. Port intention recognition to second domain.

Phase 3 Analyze annotation results. Finish intention-recognition implementation. Write thesis.

6 Conclusion

We have presented a collaborative problem-solving model which precisely defines communicative intentions and models a much wider array of collaboration than previous research. The model covers the range of collaboration paradigms as well as collaboration about (possibly interleaved) planning and acting.

We have also proposed to build an intention-recognition system based on this model for use in an implemented dialogue system and described our preliminary work on building the system.

References

[Albrecht *et al.*, 1998] David W. Albrecht, Ingrid Zukerman, and Ann E. Nicholson, "Bayesian Models for Keyhole Plan Recognition in an Adventure Game," *User Modeling and User-Adapted Interaction*, 8:5–47, 1998.

[Allen *et al.*, 2000] J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent, "An Architecture for a Generic Dialogue Shell," *Journal of Natural Language Engineering special issue on Best Practices in Spoken Language Dialogue Systems Engineering*, 6(3):1–16, December 2000.

[Allen, 1983] James Allen, "Recognizing Intentions from Natural Language Utterances," In M. Brady and R. C. Berwick, editors, *Computational Models of Discourse*, pages 107–166. MIT Press, 1983.

[Allen, 1997] James Allen, "A Problem Solving Model for Mixed-Initiative Planning," Unpublished TRAINS Technical Note, April 1997.

[Allen *et al.*, 2002] James Allen, Nate Blaylock, and George Ferguson, "A Problem Solving Model for Collaborative Agents," In *First International Joint Conference on Autonomous Agents and Multiagent Systems*, Bologna, Italy, July 15-19 2002, To appear.

[Allen *et al.*, 2001a] James Allen, George Ferguson, and Amanda Stent, "An Architecture for More Realistic Conversational Systems," In *Proceedings of Intelligent User Interfaces 2001 (IUI-01)*, pages 1–8, Santa Fe, NM, January 2001.

[Allen, 1979] James F. Allen, "A Plan-Based Approach to Speech Act Recognition," Technical Report 131/79, University of Toronto, 1979, PhD thesis.

[Allen *et al.*, 2001b] James F. Allen, Donna K. Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent, "Towards Conversational Human-Computer Interaction," *AI Magazine*, 22(4):27–37, 2001.

[Allen and Perrault, 1980] James F. Allen and C. Raymond Perrault, "Analyzing Intention in Utterances," *Artificial Intelligence*, 15(3):143–178, 1980.

[Ardissono *et al.*, 1996] L. Ardissono, G. Boella, and L. Lesmo, "Recognition of Problem-Solving Plans in Dialogue Interpretation," In *Proceedings of the Fifth International Conference on User Modeling*, pages 195–197, Kailua-Kona, Hawaii, January 1996.

[Austin, 1962] J. L. Austin, *How to Do Things with Words*, Harvard University Press, Cambridge, Massachusetts, 1962.

[Bauer, 1994] M. Bauer, "Integrating Probabilistic Reasoning into Plan Recognition," In A. Cohn, editor, *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 620–624, Amsterdam, Netherlands, August 1994. John Wiley & Sons.

[Blaylock, 2001] Nate Blaylock, "Retroactive Recognition of Interleaved Plans for Natural Language Dialogue," Technical Report 761, University of Rochester, Department of Computer Science, December 2001.

[Burmeister and Sundermeyer, 1992] Birgit Burmeister and Kurt Sundermeyer, "Cooperative Problem-Solving Guided by Intentions and Perception," In Eric Werner and Yves Demazeau, editors, *Decentralized A.I. 3: Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Kaiserslautern, Germany, August 5–7, 1991*, pages 77–92. Elsevier Science Publishers, Amsterdam, 1992.

[Calistri-Yeh, 1991] Randall J. Calistri-Yeh, "Utilizing User Models to Handle Ambiguity and Misconceptions in Robust Plan Recognition," *User Modeling and User-Adapted Interaction*, 1(4):289–322, 1991.

[Carberry, 1987] Sandra Carberry, "Pragmatic Modeling: Toward a Robust Natural Language Interface," *Computational Intelligence*, 3:117–136, 1987.

[Carberry, 1990a] Sandra Carberry, "Incorporating Default Inferences into Plan Recognition," In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 471–478, Boston, July 29 – August 3 1990. AAAI.

[Carberry, 1990b] Sandra Carberry, *Plan Recognition in Natural Language Dialogue*, ACL-MIT Press Series on Natural Language Processing. MIT Press, 1990.

[Carberry *et al.*, 1992] Sandra Carberry, Zunaid Kazi, and Lynn Lambert, "Modeling Discourse, Problem-Solving and Domain Goals Incrementally in Task-Oriented Dialogue," In *Proc. 3rd Int. Workshop on User Modeling*, pages 192–201. Wadern, 1992.

[Charniak and Goldman, 1991] Eugene Charniak and Robert Goldman, "A Probabilistic Model for Plan Recognition," In *Proc. 9th Conf. AAAI*, pages 160–165, Anaheim, CA, USA, 1991.

[Charniak and Goldman, 1993] Eugene Charniak and Robert P. Goldman, "A Bayesian Model of Plan Recognition," *Artificial Intelligence*, 64(1):53–79, 1993.

[Chu-Caroll and Brown, 1997] Jennifer Chu-Caroll and Michael K. Brown, "Initiative in Collaborative Interactions — Its Cues and Effects," In S. Haller and S. McRoy, editors, *Working Notes of AAAI Spring 1997 Symposium on Computational Models of Mixed Initiative Interaction*, pages 16–22, Stanford, CA, 1997.

[Chu-Carroll and Carberry, 1994] Jennifer Chu-Carroll and Sandra Carberry, "A Plan-Based Model for Response Generation in Collaborative Task-Oriented Dialogues," In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 799–805, Seattle, WA, 1994.

[Chu-Carroll and Carberry, 1995] Jennifer Chu-Carroll and Sandra Carberry, "Communication for Conflict Resolution in Multi-Agent Collaborative Planning," In V. Lesser, editor, *Proceedings of the First International Conference on Multiagent Systems*, pages 49–56. AAAI Press, 1995.

[Chu-Carroll and Carberry, 1996] Jennifer Chu-Carroll and Sandra Carberry, "Conflict Detection and Resolution in Collaborative Planning," In M. Woodbridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II: Agent Theories, Architectures, and Languages*,

number 1037 in Lecture Notes in Artificial Intelligence, pages 111–126. Springer-Verlag, 1996.

[Chu-Carroll and Carberry, 2000] Jennifer Chu-Carroll and Sandra Carberry, "Conflict Resolution in Collaborative Planning Dialogues," *International Journal of Human-Computer Studies*, 53(6):969–1015, 2000.

[Cohen, 1978] Philip R. Cohen, "On Knowing What to Say: Planning Speech Acts," Technical Report 118, Department of Computer Science, University of Toronto, Ontario, January 1978, PhD thesis.

[Cohen and Levesque, 1990] Philip R. Cohen and Hector J. Levesque, "Intention is Choice with Commitment," *Artificial Intelligence*, 42:213–261, 1990.

[Cohen and Perrault, 1979] Philip R. Cohen and C. Raymond Perrault, "Elements of a Plan-Based Theory of Speech Acts," *Cognitive Science*, 3:177–212, 1979.

[Cohen *et al.*, 1982] Philip R. Cohen, C. Raymond Perrault, and James F. Allen, "Beyond Question Answering," In Wendy G. Lehnert and Martin H. Ringle, editors, *Strategies for Natural Language Processing*, pages 245–274. Lawrence Erlbaum Associates, 1982.

[Cohen *et al.*, 1994] Robin Cohen, Ken Schmidt, and Peter van Beek, "A Framework for Soliciting Clarification from Users during Plan Recognition," In *Proceedings of the Fourth International Conference on User Modeling*, Hyannis, Massachusetts, August 1994.

[Ferguson and Allen, 1998] George Ferguson and James F. Allen, "TRIPS: An Intelligent Integrated Intelligent Problem-Solving Assistant," In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 567–573, Madison, WI, July 1998.

[Ferguson *et al.*, 1996] George M. Ferguson, James F. Allen, Brad W. Miller, and Erik K. Ringger, "The Design and Implementation of the TRAINS-96 System: A Prototype Mixed-Initiative Planning Assistant," TRAINS Technical Note 96-5, University of Rochester, Department of Computer Science, October 1996.

[Goldman, 1990] Robert Prescott Goldman, "A Probabilistic Approach to Language Understanding," Technical Report CS-90-34, Department of Computer Science, Brown University, December 1990, PhD thesis.

[Grice, 1957] H. Paul Grice, "Meaning," *Philosophical Review*, 66(3):377–388, 1957.

[Grice, 1969] H. Paul Grice, "Utterer's Meaning and Intention," *Philosophical Review*, 78(2):147–177, 1969.

[Grice, 1975] H. Paul Grice, "Logic and Conversation," In P. Cole and J. L. Morgan, editors, *Speech Acts*, volume 3 of *Syntax and Semantics*, pages 41–58. Academic Press, New York, 1975.

[Grosz and Sidner, 1986] Barbara Grosz and Candace Sidner, "Attention, Intention, and the Structure of Discourse," *Computational Linguistics*, 12(3):175–204, 1986.

[Grosz, 1981] Barbara J. Grosz, "Focusing and Description in Natural Language Dialogues," In A. Joshi, B. Webber, and I. Sag, editors, *Elements of Discourse Understanding*, pages 84–105. Cambridge University Press, New York, New York, 1981.

[Grosz and Kraus, 1996] Barbara J. Grosz and Sarit Kraus, "Collaborative Plans for Complex Group Action," *Artificial Intelligence*, 86(2):269–357, 1996.

[Grosz and Kraus, 1999] Barbara J. Grosz and Sarit Kraus, "The Evolution of SharedPlans," In A. Rao and M. Wooldridge, editors, *Foundations and Theories of Rational Agency*, pages 227–262. Kluwer, 1999.

[Grosz and Sidner, 1990] Barbara J. Grosz and Candace L. Sidner, "Plans for Discourse," In P. R. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 417–444. MIT Press, Cambridge, MA, 1990.

[Hinkelman and Allen, 1989] Elizabeth A. Hinkelman and James F. Allen, "Two Constraints on Speech Act Ambiguity," In *Proceedings of the Association for Computational Linguistics (ACL)*, Vancouver, Canada, 1989.

[Horvitz and Paek, 1999] Eric Horvitz and Tim Paek, "A Computational Architecture for Conversation," In *Proceedings of the Seventh International Conference on User Modeling, Banff Canada*, pages 201–210. Springer-Verlag, June 1999.

[Horvitz and Paek, 2000] Eric Horvitz and Tim Paek, "DeepListener: Harnessing Expected Utility to Guide Clarification Dialog in Spoken Language Systems," In *Proceedings of the 6th International Conference on Spoken Language Processing*, Beijing, China, 2000.

[Jennings *et al.*, 2001] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge, "Automated Negotiation: Prospects, Methods and Challenges," *International Journal of Group Decision and Negotiation*, 10(2):199–215, 2001.

[Kautz, 1990] H. Kautz, "A Circumscriptive Theory of Plan Recognition," In P. R. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 105–134. MIT Press, Cambridge, MA, 1990.

[Kautz, 1991] Henry Kautz, "A Formal Theory of Plan Recognition and its Implementation," In J. Allen, H. Kautz, R. Pelavin, and J. Tenenberg, editors, *Reasoning about Plans*, pages 69–125. Morgan Kaufman, San Mateo, CA, 1991.

[Kautz and Allen, 1986] Henry Kautz and James Allen, "Generalized Plan Recognition," In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 32–37, Philadelphia, 1986.

[Kautz, 1987] Henry A. Kautz, "A Formal Theory of Plan Recognition," Technical Report 215, University of Rochester, Department of Computer Science, 1987, PhD thesis.

[Lambert, 1993] Lynn Lambert, "Recognizing Complex Discourse Acts: A Tripartite Plan-Based Model of Dialogue," Technical Report 93-19, University of Delaware, Department of Computer and Information Sciences, Newark, Delaware, May 1993, PhD thesis.

[Lambert and Carberry, 1991] Lynn Lambert and Sandra Carberry, "A Tripartite Plan-based Model of Dialogue," In *Proceedings of the 29th ACL*, pages 47–54, Berkeley, CA, June 1991.

[Lesh, 1997] Neal Lesh, "Adaptive Goal Recognition," In *IJCAI '97*, 1997.

[Lesh, 1998] Neal Lesh, *Scalable and Adaptive Goal Recognition*, PhD thesis, University of Washington, 1998.

[Lesh and Etzioni, 1995] Neal Lesh and Oren Etzioni, "A Sound and Fast Goal Recognizer," In *IJCAI95 - Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1704–1710, Montreal, Canada, 1995.

[Lesh and Etzioni, 1996] Neal Lesh and Oren Etzioni, "Scaling up Goal Recognition," In *KR-96 - Principles of Knowledge Representation and Reasoning*, pages 178–189, 1996.

[Lesh *et al.*, 1999] Neal Lesh, Charles Rich, and Candace L. Sidner, "Using Plan Recognition in Human-Computer Collaboration," In *Proceedings of the Seventh International Conference on User Modeling, Banff Canada*. Springer-Verlag, June 1999, Also available as MERL Tech Report TR-98-23.

[Levesque *et al.*, 1990] H. Levesque, P. Cohen, and J. Nunes, "On Acting Together," In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 94–99, Boston, July 29 – August 3 1990. AAAI.

[Litman, 1985] Diane J. Litman, "Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues," Technical Report TR170, University of Rochester, Department of Computer Science, 1985, PhD thesis.

[Litman, 1986] Diane J. Litman, "Understanding Plan Ellipsis," In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 619–624, Philadelphia, 1986.

[Litman and Allen, 1987] Diane J. Litman and James F. Allen, "A Plan Recognition Model for Subdialogues in Conversations," *Cognitive Science*, 11(2):163–200, 1987.

[Litman and Allen, 1990] Diane J. Litman and James F. Allen, "Discourse Processing and Commonsense Plans," In P. R. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 365–388. MIT Press, Cambridge, MA, 1990.

[Lochbaum, 1991] Karen E. Lochbaum, "An Algorithm for Plan Recognition in Collaborative Discourse," In *Proceedings of the 29th ACL*, pages 33–38, Berkeley, CA, June 1991.

[Lochbaum, 1998] Karen E. Lochbaum, "A Collaborative Planning Model of Intentional Structure," *Computational Linguistics*, 24(4):525–572, 1998.

[Lochbaum *et al.*, 2000] Karen E. Lochbaum, Barbara J. Grosz, and Candace L. Sidner, "Discourse Structure and Intention Recognition," In Robert Dale, Hermann Moisl, and Harold Sommers, editors, *Handbook of Natural Language Processing*, pages 123–146. Marcel Dekker, New York, 2000.

[Lochbaum, 1994] Karen Elizabeth Lochbaum, "Using Collaborative Plans to Model the Intentional Structure of Discourse," Technical Report TR-25-94, Harvard University, Center for Research in Computing Technology, 1994, PhD thesis.

[Mayfield, 1992] James Mayfield, "Controlling Inference in Plan Recognition," *User Modeling and User-Adapted Interaction*, 2(1-2):55–82, 1992.

[McCarthy, 1979] 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.

[Osawa and Tokoro, 1992] Ei-Ichi Osawa and Mario Tokoro, "Collaborative Plan Construction for Multiagent Mutual Planning," In Eric Werner and Yves Demazeau, editors, *Decentralized A.I. 3: Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Kaiserslautern, Germany, August 5–7, 1991*, pages 169–187. Elsevier Science Publishers, Amsterdam, 1992.

[Paek and Horvitz, 2000] Tim Paek and Eric Horvitz, "Conversation as Action Under Uncertainty," In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, Stanford, CA, June 2000.

[Paek *et al.*, 2000] Tim Paek, Eric Horvitz, and Eric Ringger, "Continuous Listening for Unconstrained Spoken Dialogue," In *Proceedings of the 6th International Conference on Spoken Language Processing*, Beijing, China, 2000.

[Pollack, 1990] M. E. Pollack, "Plans as Complex Mental Attitudes," In P. R. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 77–103. MIT Press, Cambridge, MA, 1990.

[Pollack, 1986a] Martha Pollack, "Inferring Domain Plans in Question-Answering," Technical Report MS-CIS-86-40 LINC LAB 14, University of Pennsylvania, May 1986, PhD thesis.

[Pollack, 1986b] Martha E. Pollack, "A Model of Plan Inference that Distinguishes between the Beliefs of Actors and Observers," In *Proceedings of the Twenty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 207–214, New York, June 1986.

[Pollack, 1987] Martha E. Pollack, "Some Requirements for a Model of the Plan Inference Process in Conversation," In Ronan Reilly, editor, *Communication Failure in Dialogue*, pages 245–256. North-Holland, Amsterdam, 1987.

[Quilici, 1989] Alexander Quilici, "Detecting and Responding to Plan-Oriented Misconceptions," In Alfred Kobsa and Wolfgang Wahlster, editors, *User Models in Dialog Systems*, pages 108–132. Springer-Verlag, Berlin, 1989.

[Ramshaw, 1989a] Lance A. Ramshaw, "A Metaplan Model for Problem-Solving Discourse," In *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, pages 35–42, Manchester, England, 1989.

[Ramshaw, 1991] Lance A. Ramshaw, "A Three-Level Model for Plan Exploration," In *Proceedings of the 29th ACL*, pages 39–46, Berkeley, CA, June 1991.

[Ramshaw, 1989b] Lance Arthur Ramshaw, "Pragmatic Knowledge for Resolving Ill-Formedness," Technical Report 89-18, University of Delaware, Newark, Delaware, June 1989, PhD thesis.

[Rao and Georgeff, 1991] Anand S. Rao and Michael P. Georgeff, "Modeling Rational Agents within a BDI-Architecture," In James Allen, Richard Fikes, and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning*, pages 473–484, Cambridge, Massachusetts, April 22-25 1991. Morgan Kaufmann.

[Rich and Sidner, 1997] Charles Rich and Candace L. Sidner, "COLLAGEN: When Agents Collaborate with People," In *First International Conference on Autonomous Agents*, pages 284–291, Marina del Rey, CA, February 1997, Also available as MERL Technical Report TR96-16 and also reprinted in M. Huhns and M. Singh, editors, Readings in Agents, Morgan Kaufmann, San Francisco, CA, 1998, pp. 117–124.

[Rich and Sidner, 1998] Charles Rich and Candace L. Sidner, "COLLAGEN: A Collaboration Manager for Software Interface Agents," *User Modeling and User-Adapted Interaction*, 8(3–4):315–350, 1998, Also available as MERL Technical Report 97-21a.

[Rich *et al.*, 2001] Charles Rich, Candace L. Sidner, and Neal Lesh, "COLLAGEN: Applying Collaborative Discourse Theory to Human-Computer Interaction," *AI Magazine*, 22(4):15–25, 2001, Also available as MERL Tech Report TR-2000-38.

[Schmidt *et al.*, 1978] C. F. Schmidt, N. S. Sridharan, and J. L. Goodson, "The Plan Recognition Problem: An Intersection of Psychology and Artificial Intelligence," *Artificial Intelligence*, 11:45–83, 1978.

[Searle, 1975] John R. Searle, "Indirect Speech Acts," In P. Cole and J. L. Morgan, editors, *Speech Acts*, volume 3 of *Syntax and Semantics*, pages 59–82. Academic Press, New York, 1975.

[Searle, 1990] John R. Searle, "Collective Intentions and Actions," In P. R. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 401–415. MIT Press, Cambridge, MA, 1990.

[Sidner, 1994a] Candace L. Sidner, "An Artificial Discourse Language for Collaborative Negotiation," In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 814–819, Seattle, WA, 1994, Also available as Lotus Technical Report 94-09.

[Sidner and Israel, 1981] Candace L. Sidner and David J. Israel, "Recognizing Intended Meaning and Speakers' Plans," In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 203–208, Vancouver, B.C., 1981.

[Sidner, 1994b] Candy Sidner, "Negotiation in Collaborative Activity: A Discourse Analysis," *Knowledge-Based Systems*, 7(4):265–267, 1994, Also available as Lotus Technical Report 94-10.

[Traum, 1994] David R. Traum, "A Computational Theory of Grounding in Natural Language Conversation," Technical Report 545, University of Rochester, Department of Computer Science, December 1994, PhD Thesis.

[van Beek and Cohen, 1991] Peter van Beek and Robin Cohen, "Resolving Plan Ambiguity for Cooperative Response Generation," In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 938–944, Australia, 1991.

[Van Wie, 2001] Michael Van Wie, *Role Selection in Teams of Non-Communicating Agents*, PhD thesis, University of Rochester, Department of Computer Science, 2001.

[Wooldridge and Jennings, 1994] M. Wooldridge and N. R. Jennings, "Formalizing the Cooperative Problem Solving Process," In *Proceedings of the 13th International Workshop on Distributed Intelligence*, pages 403–417, Lake Quinalt, WA, 1994.

[Wooldridge and Jennings, 1996] Michael Wooldridge and Nicholas R. Jennings, "Towards a Theory of Cooperative Problem Solving," In John W. Perram and Jean-Pierre Muller, editors, *Distributed Software Agents and Applications: 6th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Maamaw'94, Odense, Denmark*, number 1069 in Lecture Notes in Artificial Intelligence. Springer-Verlag, 1996.

[Wooldridge and Jennings, 1999] Michael Wooldridge and Nicholas R. Jennings, "The Cooperative Problem-Solving Process," *Journal of Logic and Computation*, 9(4):563–592, 1999.